Manuel Utilisateur

24 janvier 2022

Table des matières

1 Commandes du compilateur Deca

Le compilateur Deca va générer un fichier .ass de votre fichier .deca. Une fois ce fichier créé, pour avoir l'exécutable il faudra utiliser ima.
Pour lancer le compilateur deca sur un fichier, il suffit de faire : decac fichier.deca
Une fois ceci fait, pour obtenir l'exécutable il faudra faire : ima fichier.ass

Il est également possible de lancer le compilateur sur plusieurs fichiers deca en série comme suit : decac fichier1.deca fichier2.deca etc...

Pour compiler ces fichiers en parallèle on pourra utiliser la commande :
decac -P fichier1.deca fichier2.deca etc...

Pour afficher la bannière de l'équipe de développement, utiliser la commande : decac -b

Pour arrêter la compilation après la construction de l'arbre est l'afficher, utiliser la commande : decac -p fichier.deca

Pour arrêter la compilation après l'étape de vérification (pas de sortie s'il n'y a pas d'erreur), utiliser la commande : decac -v fichier.deca

Pour limiter le nombre de registres disponibles (de R0 à RX), utiliser la commande : decac -r X

Pour activer les traces de débogage, utiliser le commande (il est possible de l'utiliser plusieurs fois pour avoir plus de traces) : decac -d fichier.deca

Pour activer les commentaires dans les fichiers .ass générés, utiliser la commande : decac -c fichier.deca

Pour afficher le mannuel de commande pour decac, utiliser simplement : decac

2 Limitations et points propres

Dans certains cas, les opérations logiques testent les deux paramètres (gauche et droit) avant de déterminer le résultat, il est plus efficace de renvoyer le résultat directement lorsque l'on sait que la valeur de droite ne changera rien au résultat (exemple : true ou ? retournera forcément true). Cela laisserait place à des manières de coder plus optimales.

Nous avions implémenté la gestion des caractères spéciaux dans les prints mais nous avons du la retirer lors du suivi2 car il nous a été dit que cette fonctionnalité n'étant pas dans la spécification, il ne fallait absolument pas l'implémenter. Cela nous a valu des points en moins... Nous pensions qu'une fonctionnalité en plus serait bénéfique étant donné que cela rajoute des possiblités à l'utilisateur en plus d'être plus pratique. Ceci n'est donc plus disponible.

Nous avons tenté d'utiliser l'instruction assembleur FMA pour optimiser le code des fichiers assembleurs générés par decac mais nous n'avons pas réussi. Cela aurait amélioré l'efficacité de notre compilateur.

Nous avons ajouté l'option -c en plus, décrite ci-dessus dans la partie commandes du compilateur.
Mais nous n'avonspas implémenté la commande -w.

3 Messages d'erreur

3.1 Erreurs Lexicales et Syntaxiques

Nous listerons toutes les erreurs lexicales et sy,taxiques qu'il est possible de rencontrer avec notre compilateur deca, ainsi que les raisons pour lesquelles cela peut se produire.

- No viable alternative for TOKEN. : Erreur rencontrée lorsqu'un symbole n'est pas reconnu par le lexer (token recognition error).

- Circular include for file FILE. : Erreur rencontrée lorsqu'un fichier s'inclut lui-même (de manière directe ou indirecte).

- Float is too large. : Erreur rencontrée lorsqu'une valeur trop grande est affectée à un flottant.

- FICHIER : include file not found. : Erreur rencontrée lorsqu'un fichier que l'on tente d'inclure n'est pas trouvé (fichier inexistant ou mauvais chemin).

- Int is too large. : Erreur rencontrée lorsqu'une valeur trop grande est affectée à un entier.

- left-hand side of assignment is not an lvalue. : Erreur rencontrée lorsque l'on tente d'affecter une valeur à quelque chose qui n'est pas une variable.

3.2 Erreurs contextuelles

Nous listerons ici toutes les erreurs de type Contextual error qu'il est possible de rencontrer avec notre compilateur deca, ainsi que les raisons pour lesquelles cela peut se produire.

- Can't assign TYPE to ExpectedType : Erreur rencontrée lorsque le type d'une expression binaire ne correspond pas au type attendu et ne peut pas être converti en celui ci.

-Condition must be boolean : TYPE given : Erreur rencontrée lorsqu'une condition n'est pas de type boolean.

- Invalid argument for arithmetic operation. Args should be of type [int, float] : left operand type and right operand type given. : Erreur rencontrée lorsque l'un des arguments d'une opération arithmétique n'est pas de type int ou float.

- Failed to convert the right Operand to Float ou Failed to convert the left Operand to Float : Cette erreur est rencontrée lorsque, respectivement, la partie droite ou gauche d'une opération arithmétique ou d'une comparaison n'a pas pu être convertie en Float.

- Invalid argument for boolean operation. Args should be boolean : TYPE given. : Erreur rencontrée lorsqu'un argument d'une opération booléenne n'est pas de type boolean.

- Invalid argument for comparison operation. Args should be of type [int, float] : left operand type and right operand type given. : Erreur rencontrée lorsque la partiedroite et/ou gauche d'une comparaison n'est pas de type int ou float.

- Invalid argument to function print(ln). Args should be of type [int, float, String] : TYPE given. : Erreur rencontrée lorsque le/les argument(s) d'une fonction de type print() n'est/ne sont pas de type int, float ou string.

- Impossible to cast TYPE to castType. : Erreur rencontrée lorsque l'on tente de faire un cast invalide, pour lequel il est impossible de convertir le type de l'expression/variable vers le type du cast.

- Invalid argument for conversion of an int into a float. Arg should be an int : TYPE given. : Erreur rencontrée lorsque l'on tente de convertir un entier en float mais que l'argument donné n'est pas de type int.

- Class CLASSE is already declared. : Erreur rencontrée lorsque l'on déclare une classe déjà existante (ici CLASSE).

- Type cannot be void. : Erreur rencontrée lorsque l'on tente de déclarer une variable de type void.

- Field FIELD already exists. : Erreur rencontrée lorsque l'on tente de déclarer un champ déjà existant dans une classe (ici FIELD).

- Wrong type, expecting int or float matrix, given matrixType. : Erreur rencontrée lorsque l'on tente de déclarer une matrice d'un autre type que int ou float.

- Too much arguments to declare a matrix, expected 1 or 2, given : X. : Erreur rencontrée lorsque l'on tente de déclarer une matrice avec plus de 2 arguments.

- Trying to declare a matrix with size of type TYPE, int required. : Erreur rencontrée lorsque l'on tente de déclarer la taille d'une matrice avec un type autre qu'un int.

- DeclMatrix MATRIX already exists. : Erreur rencontrée lorsque l'on tente de déclarer une matrice déjà existante.

- Method METHODE already exists. : Erreur rencontrée lorsque l'on tente de déclarer une méthode déjà existante dans la classe en question.

- Trying to override a field and not a method : FIELD. : Erreur rencontrée lorsque l'on tente de redéfinir une champ et non une méthode.

- Override impossible : return type is not the same. : Erreur rencontrée lorsque l'on tente de redéfinir une méthode existante avec le mauvais type de retour.

- Override impossible : Signature is not the same. : Erreur rencontrée lorsque l'on tente de redéfinir une méthode existante avec de mauvais arguments.

- Parameter type can't be null or void. : Erreur rencontrée lorsque l'on tente de passer un paramètre de type void ou null à une méthode.

- PARAM is already defined. : Erreur rencontrée lorsque l'on tente de rajouter un paramètre déjà existante dans l'environnement en question.

- Variable can not be of type void. : Erreur rencontrée lorsque l'on tente de déclarer une variable de type void.

- Variable VAR is already declared. : Erreur rencontrée lorsque l'on tente de déclarer une variable déjà existante dans l'environnement en question.
- Cannot apply dot to something that is not a class : OBJECT given. : Erreur rencontrée lorsque l'on tente d'accéder au champ d'un objet qui n'est pas une classe avec le symbole ".".

- Field FIELD doesn't exist in class. : Erreur rencontrée lorsque l'on tente d'accéder à un champ qui n'existe pas dans une classe.

- Field FIELD is PROTECTED. : Erreur rencontrée lorsque l'on tente d'accéder à un champ protected s'une classe.

- Variable VAR is undefined. : Erreur rencontrée lorsque l'on tente d'utiliser une variable non définie. :

- Type TYPE is undefined. : Erreur rencontrée lorsque l'on tente de déclarer une variable d'un type non défini.

- OBJECT doesn't exist. et TYPE isn't a type. : Erreurs rencontrées lorsque l'on tente de faire un instanceof sur un objet non défini et/ou avec un type non défini.

- Too much arguments to access a matrix, expected 1 or 2, given : X. : Erreur rencontrée lorsque l'on donne trop d'argument pour accéder à une valeur d'une matrice.

- Trying to access a matrix with index of type TYPE, int required. : Erreur rencontrée lorsque l'on tente d'accéder à une valeur de matrice en donnant un indice d'un autre type qu'int.

- Cannot access one dimensional array with two args ou Cannot access two dimensional array with one args. : Erreurs rencontrées respectivement lorsque l'on tente d'accéder aux valeurs d'un matrice à une dimension avec deux indices et lorsque l'on tente d'accéder aux valeurs d'une matrice à deux dimensions avec seulement un indice.

- Cannot apply method to something that is not a class : TYPE given. : Erreur rencontrée lorsque l'on tente d'appliquer une méthode avec le "." à une variable qui n'est pas une classe.

- Method not declared : METHODE. : Erreur rencontrée lorsque l'on tente d'accéder à une méthode non définie.

- Incompatible arguments to method METHODE : given ARGS, expected : ARGS. : Erreur rencontrée lorsuque l'on tente d'appeler une méthode avec les mauvais arguments.

- Invalid argument for modulo operation. Both args should be int. TYPE and TYPE given. : Erreur encontrée lorsque l'on tente de faire un modulo (%) avec des types autres que des entiers.

- new can't be used to declare type : TYPE. : Erreur rencontrée lorsque l'on tente de faire un new sur un type qui n'est pas une classe.

- Invalid argument for not operation. Arg should be of type [boolean]. : Erreur rencontrée lorsque l'on tente d'utiliser l'opération not (!) sur une variable ou valeur qui n'est pas de type boolean.

- Invalid argument to condition in return statement. Return type does not match with type provided : TYPE given, TYPE expected. : Erreur rencontrée lorsque l'on tente de retourner une valeur ou variable qui ne correspond pas au type de retour spécifié pour la méthode en question.

- Cannot use 'this' in main program. : Erreur rencontrée lorsque l'on utilise l'instruction this dans la fonction main.

- This is not in a class declaration. : Erreur rencontrée lorsque l'on utilise l'instruction this en dehors d'un environnement de classe.

- Invalid argument for unary minus operation. Arg should be type of [int, float] : TYPE given. : Erreur rencontrée lorsque l'on utilise l'opération unaire '-' sur une variable ou valeur qui n'est pas de type int ou float.

- Only width can be accessed on a one dimensional array, FIELD given. : Erreur rencontrée lorsque l'on tente d'accéder à un autre champ que width pour un tableau de cette manière, tab.FIELD ;

- Only width and length can be accessed on a two dimensional matrix, FIELD. : Erreur rencontrée lorsque l'on tente d'accéder à un autre champ que width ou length pour une matrice de cette manière, matrix.FIELD ;

4 Extension de la bibliothèque standard : Tableaux et calculs matriciels

Nous proposons une extension de la bibliothèque standard permettant l'utilisation de tableaux et matrices.

4.1 Accès à l'extension

Pour utiliser l'extension, il suffit d'inclure la librairie comme suit :
#include "NumDeca.decah".

4.2 Tableaux

Nous indiquerons ici les commandes relatives à l'utilisation des tableaux.
Les exemples sont donnés pour des tableaux de type int mais il est également possible d'utiliser des tableaux de type float.

Pour déclarer un tableau : int[TAILLE] Tableau ;
Il est obligatoire de fournir la Taille du tableau pour la création.
Il est également possible de faire : int[TAILLE1] tab1, [TAILLE2] tab2 ;
Toutes les valeurs du tableau sont initialisées à zéro au départ.
Pour affecter des valeurs aux différents indices d'un tableau il est nécessaire d'affecter une valeur indice par indice. Pour parcourir les tableaux, libre à vous d'utiliser une boucle while.
Pour accéder à une valeur d'un tableau :
- Affecter : tab[INDICE] = valeur ;
- Récupérer : var = tab[INDICE] ; ou encore directement dans une méthode ou un print.
Pour affecter un tableau à un autre (ils doivent faire la même taille) :
int[2] t1, [2] t2 ;
t1 = t2 ;
Pour récupérer la taille d'un tableau : taille = tab.width ;

4.3 Matrices

Nous indiquerons ici les commandes relatives à l'utilisation des matrices.
Les exemples sont donnés pour des matrices de type int mais il est également possible d'utiliser des matrices de type float.

4.3.1 Utilisation générale

Pour déclarer une matrice : int[LIGNE, COL] Matrice ;
Il est obligatoire de fournir les Tailles de la matrice pour la création.
Il est également possible de faire : int[LIGNE1, COL1] Mat1, [LIGNE2, COL2] Mat2 ;
Toutes les valeurs sont initialisées à zéro au départ.
Pour affecter des aux différents indices d'une matrice il est nécessaire d'affecter une valeur indice par indice. Pour parcourir les matrices, libre à vous d'utiliser deux boucles while imbriquées.
Pour accéder à une valeur d'une matrice :
- Affecter : mat[i,j] = valeur ;
- Récupérer : var = mat[i,j] ; ou directement dans une méthode ou un print.
Pour affecter une matrice à une autre (elles doivent faire la même taille) :
int[2,2] m1, [2,2] m2 ;
m1 = m2 ;
Pour récupérer les dimensions d'une matrice :
longueur = mat.length ;
largeur = mat.width ;

4.3.2 Fonctions sur les matrices

Méthodes de la classe MatrixSet

void floatSetIdentity(float[] matrix) ;
void intSetIdentity(int[] matrix) ;
Transforme la matrice passée en paramètre (cette matrice doit être carrée) en la matrice identité de taille correspondante.

void intSetZero(int[] matrix) ;
void floatSetZero(float[] matrix) ;
Met toutes les valeurs de la matrice passée paramètre à zéro.

void intSetOne(int[] matrix) ;
void floatSetOne(float[] matrix) ;
Met toutes les valeurs de la matrice passée paramètre à un.

Pour utiliser ces méthodes de la classe MatrixSet il est nécessaire de créer une instance de cette classe, par exemple :
MatrixSet ms = new MatrixSet() ;
ms.intSetIdentity(int[5,5] matrix) ;

Méthodes de la classe MatrixOperations

void intSum(int[] mat1, int[] mat2) ;
void floatSum(float[] mat1, float[] mat2) ;
Modifie mat1 en la somme des deux matrices passées en paramètre (les deux matrices doivent avoir les mêmes dimensions).

void intMultConst(int[] matrix, float const) ;
void floatMultConst(flaot[] matrix, float const) ;
Modifie la matrice matrix en le produit de celle-ci et de la constante const.

void intTranspose(int[] matrix, int[] output) ;
void floatTranspose(float[] matrix, float[] output) ;
Stocke la transposée de matrix dans la matrice output. Si celle ci est de taille n, p, la transposée sera de taille p, n.

void intMult(int[] mat1, int[] mat2) ;
void floatMult(float[] mat1, float[] mat2) ;
Retourne la matrice produit des deux matrices passées en paramètre (mat1 doit avoir son paramètre COL1 égal au paramètre LIGNE2 de mat2, la matrice retournée sera de taille LIGNE1, COL2).

Pour utiliser ces méthodes de la classe MatrixOperations il est nécessaire de créer une instance de cette classe, par exemple :
MatrixOperations mo = new MatrixOperations() ;
int[5,5] m ;
m = mo.intSum(int[5,5] matrix1, int[5,5] matrix2) ;

Méthodes de la classe MatrixUtils

boolean intMatrixEqual(int[] mat1, int[] mat2) ;
boolean floatMatrixEqual(float[] mat1, float[] mat2) ;
Retourne true si les deux matrices passées en paramètres sont egales, false sinon.

void intMatrixPrint(int[] matrix) ;
void floatMatrixPrint(float[] matrix) ;
Affiche le contenu de la matrice passée en paramètre.

void intMatrixCopy(int[] matrix, int[] output) ;
void floatMatrixCopy(float[] matrix, float[] output) ;
Copie matrix dans la matrix output.

void intToFloat(int[] matrix, float[] output) ;
Convertie la matrice matrix en matrice de type flottant et la stocke dans la matrice output.

Pour utiliser ces méthodes de la classe MatrixUtils il est nécessaire de créer une instance de cette classe, par exemple :
MatrixUtils mu = new MatrixUtils() ;
mu.intMatrixPrint(int[5,5] matrix) ;

4.4 Limitations

Nous n'avons pas implémenté de for each pour le parcours des tableaux et matrices.

Il n'est pas possible d'initialiser directement les tableaux et matrices avec les valeurs que l'on veut, il faut parcourir la totalité de la structure pour pouvoir affecter des valeurs à tous les indices.

Les fichiers DecompuLU.todo et NumDeca.todo dans src/main/resources/include/ contiennent des implémentations non fonctionnelles pour l'extension. Nous avions commencé par ces fichiers néanmoins nous ne pouvons pas retourner de matrices et avons donc changer l'implémentation.

Nous n'avons pas eu le temps de faire tout ce que nous voulions pour l'extension : Algorithme de strassen, Calcul du déterminant, Matrices inverses, Valeurs propres.
Néanmoins toutes ces fonctionnalités étaient considérées comme "nice to have", c'est pourquoi nous avions décidé de les traiter en dernier si nous avions le temps.