École nationale de l'informatique et des mathématiques appliquées de Grenoble

Projet Génie Logiciel

Année universitaire 2020-2021
Manuel Utilisateur

Groupe 8 Equipe 40
Étudiants 2A

Table des matières

1 Limitations et points propres du compilateur

1.1 Points propres du compilateur

1.1.1 Points propres du compilateur

Il est possible d'utiliser des méthodes directement écrites en assembleur à l'aide de la définition suivante :

type_retour nom_méthode(paramètres) asm(langage en assembleur) ;

Concernant, le langage en assembleur contenu entre les parenthèses il est nécessaire d'aller à la ligne à chaque fois que nécessaire, par exemple :

LOAD #1, R2
STORE R2, 1(GB)

Toute autre écriture du corps de la méthode en assembleur sera mal ajoutée dans le code assembleur général de la compilation du programme et entraînera une erreur du compilateur.
Toutes les erreurs concernant la compilation d'un programme restent disponibles lors de l'écriture d'un corps de méthode en assembleur (voir la liste d'erreurs de compilation disponible partie II-E). Par exemple, il est possible de faire appel à une erreur dans le corps de la méthode :

CMP #null, R3
BEQ dereferencement_null

1.1.2 La sélection multiple (cas des objets)

Dans le cas des objets, la sélection "à la chaîne" ne nécessite aucun parenthésage particulier. Exemple : dans le cadre d'une classe A dont une des méthodes est :

```
1 A getA(){
2     return this;
3 }
4 A a = new A();
```

Les deux expressions suivantes sont alors équivalentes :

a.getA().getA().getA() ;
(((a.getA()).getA()).getA()) ;

Ce critère de sélection est à mettre en relief avec la sélection concernant les tableaux et matrices développées par la suite.

1.2 Limites du compilateur

1.2.1 Ecriture d'une méthode dont le nom est reconnu par le lexer

Certains mots reconnus par le lexer ne peuvent pas être utilisés comme nom de méthode. En effet, en prenant l'exemple de print ou println, il est impossible d'implémenter une méthode dont le nom serait print ou println. Dans le cas d'une utilisation de tels mots comme nom de méthode une erreur de syntaxe sera transmise pour montrer l'erreur (*"no viable alternative at input 'voidprint' "* si on prend le cas où on implémente une méthode void print()).

1.2.2 Sous tableaux et déclaration partielle

Lors du développement de notre extension et du changement de la grammaire pour supporter les tableaux, la déclaration partielle de tableau ainsi que la possibilité de sélectionner un sous tableau était possible. Nous avons dû, finalement, opter pour interdire cela pour faciliter l'implémentation des tableaux pour notre langage. Ainsi ceci n'est plus possible :

```
{
    int[][][] x = new int[4][3][];
    x[2][1] = new int[1];
}
```

Le nombre d'indice pour la sélection et la création doit donc correspondre exactement à la dimension du tableau déclaré.

1.2.3 Conversion de float vers int dans les littéraux de tableaux

Lors d'une création d'un tableau sous forme de littéral :

$$int[]\ x = \{1,\ 2,\ 3\};$$

Il est impossible (par choix de conception) de mélanger *int* et *float* pour un tableau de flottant. Les éléments donnés dans un littéral doivent donc tous être du type de la déclaration.

2 Messages d'erreur

2.1 Messages d'erreur lexicographique

Message d'erreur	**Explication**
Erreur de compilation : flottant arrondi 0	Le flottant déclaré est trop petit et non nul. Il est donc arrondi vers 0 ce qui interdit. Il doit être déclaré plus grand que 0x1.0p-127 en hexadécimal ou 1.4012985E-45 en décimal.
Erreur de compilation : flottant arrondi à l'infini	Le flottant déclaré est trop grand et est arrondi à l'infini il doit être plus petit que 1.7FFFFFp127 en hexadécimal ou 3,402823 5 ×10E38 en décimal.
Erreur de compilation : Int arrondi à l'infini	Le int déclaré est trop grand, le int doit être inférieur ou égale à 2147483647
token recognition error at :	Le jeton suivant n'est pas reconnu par le compilateur.
include file not found	Le fichier à inclure n'a pas été trouvé

2.2 Messages d'erreur syntaxique

Message d'erreur	**Explication**
mismatched input 'x' expecting ...	Le token x n'est pas celui qui était attendu, les caractères proposés après le *expecting* sont ceux qui étaient attendus
Extraneous input 'x' expecting	Il manque un caractère avant le x pour que l'expression soit syntaxiquement correct. Le type de caractère attendu avant est présent après le expecting
no viable alternative at input 'x'	La syntaxe est invalide lorsqu'on met le caractère x, il n'y a pas d'alternative pour que l'expression soit reconnu avec ce caractère
missing 'x' at 'y'	Le caractère x est manquant au niveau du caractère y

2.3 Messages d'erreur contextuelle

Message d'erreur	**Explication**
This identifier is not declared in this scope	L'identifiant auquel on tente d'accéder (field, param, var, method) n'est pas défini dans cette portée. Identifiant non déclaré ou encore n'étant pas dans cette environnement mais possiblement dans un autre.
This type is not defined.	Le type demandé n'a pas été déclaré comme un type (que ce soit prédéfini ou par une classe).
The definition of this identifier is not a type type	Le type demandé n'est pas un type mais un identifiant correspondant à toute autre possibilité.
The parent class of this class was not declared before	La classe parente demandée n'existe pas (encore).
The definition of the parent class is not a class identifier	La classe parente demandée n'est pas une classe mais d'un autre type.
This class is already defined	Définition d'une classe dont le nom est déjà associé à une classe.
The class is not initialize in types	La classe demandée n'a pas été correctement déclarée précédemment.
The class is not of nature Class	La classe courante n'est pas considérée comme une classe.
The type of a field to be declared can not be void	Le type d'un champ de classe ne peut pas être void.
The field declared exists in parent class Environnement but is not a field there	Le champ déclaré pour cette classe a le même nom qu'un autre objet de la super classe qui n'est pas un champ (une méthode par exemple).
This field is already defined in this scope	Double définition d'un même champ dans une même classe
This method has the same identifier as a not method object in super class	La méthode déclarée pour cette classe a le même nom qu'un autre objet de la super classe qui n'est pas une méthode (un champ par exemple).
The redefinition of this method has not the same signature as the method defined in the super class	La méthode définie dans la courante classe a le même nom qu'une méthode de la super classe mais les deux méthodes n'ont pas une signature correspondante.
The type of return of this method is not the same as the one of the herited method	Le type de retour n'est pas le même que celui de la méthode identique dans la superclasse.

Message d'erreur	**Explication**
This method is already defined in this scope	La méthode est définie deux fois dans la même classe.
A parameter can not be void type	Un paramètre de méthode ne peut pas être un void.
This parameter is already defined for this method	Le paramètre a le nom d'un paramètre déclaré précédemment pour cette méthode.
The type of a variable to be declared can not be void	La variable déclarée ne peut pas être de type void.
The type of the variable and its assignation are not compatible	La variable et son assignation n'ont pas des types compatibles.
The type of expression to be printed is not int, float or string	L'expression à afficher à l'écran grâce à un print ne peut pas être d'un autre type que int float ou string.
At least one of the members of this operation is not compatible with this operator	L'un des membres de l'opération n'est pas compatible avec celle-ci.
The member of this unary operation is not compatible with this operator	Le membre de cette opération unaire n'est pas compatible avec celle-ci.
The nature of the left member is not of type field, parameters of variables	La nature du membre de gauche (dans le cas d'une initialisation ou d'une affectation) n'est pas un champ, un paramètre ou une variable.
The Definition of this Symbol is already in the environment	Ce symbole (nom) de variable est déjà définie.
The conversion to a float can only be from an int	La conversion vers un float a été appelée pour un objet qui n'est pas un integer.
This type is not compatible with the casted type	Les types ne sont pas compatibles pour que l'un soit cast en l'autre.
The type of the object to be instantiated is not a class	L'objet à instancier n'est pas une classe (essai de new avec un objet n'étant pas une classe).
The class pointed by this can not be Object class	This ne peut pas référencer Object.
The type of expression selected is not a class	L'objet sélectionné (lors d'un objet.qqch) n'est pas une classe.
The selected class is not a subclass of the class where the selection was made	La sélection d'un champ protégé n'est pas fait dans une sous classe de celle auquel le champ appartient.
The type of the object selected is not a subtype of the current class	Le type de l'objet sélectionné n'est pas un sous type de la classe courante.
The returnType can not be of type void	Le type de retour ne peut pas être void.
The operation InstanceOf can not be applied to those operands	Instanceof a échoué sur ces membres.

Message d'erreur	**Explication**
This identifier does not correspond to a method in this scope	Ce nom ne correspond pas à une méthode pour la visibilité dans la classe courante.
The signature of this method does not match the number of parameters given	Le nombre de paramètres lors de l'appel de cette méthode ne correspond pas à la signature de celui-ci.
This array is of null dimension : can be a base type	Le tableau créé est défini comme un tableau de dimension 0 : Il peut être vu donc comme un objet non tableau.
The base Type of this array can not be void	Le type de base d'un tableau ne peut être *void* : une déclaration *void[][][] x ;* est donc impossible.
Elements of array literal are not all of the same type	Un littéral de tableau ne peut pas contenir des éléments de type différent : *int[] x = {1, 2, "string"}*
The selection in a index can not be in a not Array object	Une sélection faite avec des crochets (*x[i]*) ne peut pas être faite sur un objet n'était pas un tableau. Par exemple, *x[i]* avec *x* un *int*.
The index of selection has to be an integer	L'indice donné dans une sélection de tableau doit être un entier. Une sélection comme ceci : *x[true], x[1.0], x["blabla"]* est donc impossible.
The dimension of the selection is different than the dimension of the object	Une sélection de tableau doit se faire avec le même nombre de dimension que celle de l'objet. Voir 1.2.2
The dimension of an array has to be give with an integer	La dimension d'un tableau lors d'un *new* ne peut être faite qu'avec un entier à l'intérieur des [].

2.4 Messages d'erreur à l'exécution

Message d'erreur	**Explication**
Erreur : debordement sur les float **(*)**	Survient lorsque les opérations (+, -, *, /) du programme deca provoquent un débordement arithmétique sur les flottants (inclut la division par 0.0).
Erreur : division par zero impossible **(*)**	Survient lorsqu'une division par zéro (entre des entiers) est mise en place dans le programme.
Erreur : reste entier par zero impossible **(*)**	Survient lorsqu'un modulo zéro est mis en place dans le programme.
Erreur : un entier est attendu	Survient lorsqu'un autre type que int est donné lors de la lecture ReadInt (il est attendu de donner un entier seulement).
Erreur : un float est attendu	Survient lorsqu'un autre type que float est donné lors de la lecture ReadFloat (il est attendu de donner un float seulement).
Erreur : V[dval] non codable sur un flottant **(*)**	Survient lors d'une tentative de conversion en flottant avec un type non valable.
Erreur : V[dval] non codable sur un entier **(*)**	Survient lors d'une tentative de conversion en entier avec un type non valable.
Erreur : debordement de pile **(*)**	Survient lorsque la pile est trop petite pour accueillir l'entièreté du programme.
Erreur : tas plein **(*)**	Survient lorsque le tas est trop petit pour accueillir l'entièreté du programme.
Erreur : dereferencement de null **(*)**	Survient lorsqu'on tente d'utiliser un objet dont la valeur est null.
Erreur : cast impossible **(*)**	Survient lorsqu'un cast non autorisé est mis en place.
Erreur : index out of range	Survient lorsque l'on veut accéder à un élément d'un tableau avec un indice trop grand ($\geqslant$ taille) ou négatif.

(*) : Ces erreurs sont supprimées avec l'option noCheck du compilateur (**-n**)

3 Mode opératoire des extensions

Nous avons choisi d'implémenter l'extension TAB pour notre compilateur. Nous avons donc ajouté au langage deca de base les outils pour déclarer des tableaux, les initialiser et accéder à leurs éléments. Nous avons aussi construit une bibliothèque de calcul matriciel, dont nous décrirons les outils.

3.1 Tableaux

3.1.1 Déclaration

Pour la déclaration de tableau, nous avons imité celle du langage Java à savoir :

```
1 Type_des_elements[] nom_du_tableau;
```

Dès lors plusieurs choses sont notables. Les éléments d'un tableau sont de même type. La taille n'est pas spécifiée à la déclaration et le tableau peut être de n'importe quelle dimension. Par exemple une matrice peut être déclarée comme suit :

```
1 int[][] matrice;
```

ou encore des tableaux de matrices :

```
1 int[][][] tab_matrice;
```

et ainsi de suite. Notons enfin qu'il est tout à fait possible que le type des éléments soit une classe :

```
1 A[] tab_classe;
```

3.1.2 Allocation

La déclaration d'un tableau ne réserve qu'une place sur la pile dédiée à l'adresse du tableau sur le tas, mais celui-ci n'a pas encore de place allouée sur le tas (même processus que la déclaration d'un objet). Pour cela il faut utiliser le mot-clé *new* :

```
1 int[] tab = new int[expr];
```

Il est à ce moment important de comprendre que expr désigne la taille du tableau, mais que expr peut être remplacé par toute expression de type *int* :

```
{
    // Les lignes suivantes produisent un tableau de meme taille (6)
    float[] tab;
    int expr = 3;
    tab = new float[expr+expr];
    tab = new float[6];
    tab = new float[2*expr];
    // ...etc
}
```

Néanmoins, comme évoqué en première partie, il est impossible de déclarer partiellement les tailles de chaque dimension. La première ligne du code suivant produit une erreur :

```
{
    float[][] tab = new float[5][]; // incorrect
    float[][] tab = new float[5][7]; // correct
}
```

L'instruction new est donc commune aux tableaux et aux objets, on pourra constater qu'ils sont gérés de manière similaire en mémoire. En effet, cette instruction alloue sur le tas la taille nécessaire pour les éléments du tableau (d'où l'impossibilité de déclarer partiellement les tailles). Il est primordial de noter qu'à ce stade les éléments ne sont pas initialisés (cf section suivante). De même que pour les variables de bases, une tentative d'accès à ces éléments provoque une erreur non répertoriée dans le tableau des erreurs d'exécution :

```
{
    float z;
    float[] tab = new float[3];
    z = tab[2]; // produira une erreur IMA
}
```

La deuxième façon d'allouer un tableau est de lui donner directement une valeur avec les conventions d'écritures décrites dans l'exemple ci dessous.

```
{
    int[] tab;
    int[][] mat;
    tab = {1, 2, 3, 4, 5};
    mat = {{1, 2}, {3, 4}};
}
```

Dans ce cas là, la mémoire sur le tas est allouée ET les valeurs sont initialisées. On peut donc y accéder avec la méthode décrite dans la section 3.1.4 Accès aux éléments d'un tableau.

3.1.3 Initialisation

Comme décrit dans la partie précédente, l'instruction *new* alloue en mémoire la place pour un tableau mais ne l'initialise pas. La méthode usuelle pour procéder est donc :

```
{
    int i, length;
    float[] tab;
    length = 5;
    tab = new float[length];
    i = 0;
    while (i < length) {
        tab[i] = i*2+1; // Exemple, on peut aussi initialiser a 0 ou autre
        i = i + 1;
    }
}
```

A noter aussi que les effets de bords sont possibles, il faut donc bien y faire attention. Considérons par exemple le cas suivant :

```
{
    int[] tab1;
    int[] tab2;
    tab1 = {1, 2, 3, 4, 5};
    tab2 = tab1;
    tab1[1] = 10;
    println(tab2[1]);
}
```

En fait, la ligne 5 ne fait pas une copie, les deux variables *tab1* et *tab2* pointent alors sur la même zone tas. Un changement sur tab1 affecte donc aussi tab2. Le programme affichera donc 10. // Pour finir, il faut noter un choix important de notre extension. Là ou la conversion des entiers en flottants étaient très permissive dans le langage sans objet, elle ne l'est plus concernant les tableaux. Voyons plutôt l'exemple :

```
{
    float z = 2; // correct, conversion automatique
    float[] tab1 = {1, 2, 3, 4}; // incorrect, produira une erreur
    float[] tab2 = {1.0, 2.0, 3.0, 4.0}; // facon correcte de coder
}
```

3.1.4 Accès aux éléments

L'accès aux éléments d'un tableau est très naturel. Les indices commencent à 0. Comme expliqué en première partie, il est impossible d'accéder partiellement à un tableau. Il est par exemple impossible de sélectionner une ligne entière d'une matrice. Il faut donc qu'une sélection

renseigne autant d'indices que la dimension du tableau. Voici un résumé dans l'exemple ci-dessous :

```
{
    float x;
    int n = 3;
    float[] tab = {1, 2, 3};
    float[][] mat = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    // instructions correctes
    x = tab[0];
    tab[n-2] = x;
    x = mat[0][n-2];
    mat[1][1] = x;
    // instructions incorrectes produisant une erreur /!\
    x = tab[n] // index out of range
    tab = mat[0] // selection partielle impossible
    mat[n-1] = tab // idem
}
```

3.2 Bibliothèques

Nous avons développé une bibliothèque de calcul contenant des classes pour les tableaux et les matrices. Pour appeler les classes et méthodes pour les tableaux, ajouter en début de fichier include "Tab.decah" pour les matrices, ajouter include "Matrice.decah"

3.2.1 Tab.decah

Cette bibliothèque (src/main/ressources/include) regroupe TabFloat.decah, TabInt.decah, Racine.decah. Elle facilite la manipulation des tableaux, nous allons lister ici les fonctionnalités proposées.
La première étape est l'initialisation. Pour cela il faut créer un tableau et l'initialiser puis le lier à l'objet. Il y a deux façons de le lier : avec ou sans copie. La première empêche les effets de bords, l'autre les permet.

```
#include "Tab.decah"
{
    float[] tab = {1.1, 2.2, 3.3};
    TabFloat t1 = new TabFloat();
    TabFloat t2 = new TabFloat();
    t1.setInitCopy(tab, 3);
    t2.setInit(tab, 3);
    tab[1] = 4.4;
    println(t1.getCase(1));
    println(t2.getCase(1));
}
```

Le premier *println* affichera 2.2 tandis que le second affichera 4.4. Ce programme est aussi l'occasion de présenter l'accès à un élément par la méthode getCase(int index).

Cette bibliothèque propose ensuite les fonctionnalités suivantes sur les objets de type TabFloat, avec les méthodes suivantes :

- *boolean add(float element, int indice)*, cette méthode ajoute l'élément *element* à l'indice *indice* du tableau. Si cet ajout n'est pas possible (index out of range) la méthode ne fait rien et renvoie false. Elle renvoie true si l'ajout se fait avec succès.

- *void addFirst(float valeur)*, ajoute l'élément *element* au début du tableau.

- *boolean setCase(int indice, float element)*, modifie un élément *element* du tableau *indice*. Renvoi false si echoue, true sinon

- *void addLast(float valeur)*, ajoute l'élément *element* en fin du tableau.

- *boolean delete(int indice)*, supprime l'élément *element* à l'indice *indice* du tableau. Renvoie false si elle échoue (mauvais indice), true sinon.

- *void deleteFirst()*, supprime le premier élément.

- *void deleteLast()*, supprime le dernier élément.

- *boolean sumTab(TabFloat tab)*, ajoute le tableau tab terme à terme à l'objet. Renvoie false si les tableaux ne sont pas de même tailles, true sinon.

- *boolean multTab(TabFloat tab)*, ajoute le tableau tab terme à terme à l'objet. Renvoie false si les tableaux ne sont pas de même tailles, true sinon.

- *void affichage()*, affiche le tableau.

- *void mergeSortAscending()*, trie le tableau par élément croissant (algorithme de tri fusion, complexité en *O(nlog(n))*, n la taille du tableau)

Tout ce qui précède est aussi utilisable pour les tableaux d'entier, avec la bibliothèque TabInt.decah. Enfin, cette liste n'est pas tout a fait exhaustive, puisque les bibliothèques contiennent d'autres méthodes, mais moins utiles (utilisées en tant que méthodes auxiliaires par les méthodes ci-dessus).

3.2.2 Matrice.decah

Cette bibliothèque (src/main/ressources/include) regroupe les fichiers Matrice.decah, AbstractMatrice.decah et MatriceFloat.decah . Le fichier Matrice.decah inclus les fichiers AbstractMatrice.decah et MatriceFloat.decah. Il existe plusieurs manières de déclarer une matrice. On passe en paramètre le tableau a deux dimensions, son nombre de lignes et son nombre de colonnes.

```
#include "Matrice.decah"
{
    float[][] m1 = {{1.0, 2.0}, {2.0,3.0}};
    int[][] m2 = {{1,2}, {2,4}};
    MatriceFloat mat1 = new MatriceFloat();
    MatriceFloat mat2 = new MatriceFloat();
    MatriceFloat mat3 = new MatriceFloat();
    MatriceFloat mat4 = new MatriceFloat();
    MatriceFloat mat5 = new MatriceFloat();
    // initialisation par reference
    mat1.setInit(m1, 2, 2);
    // initialisation par copie du tableau de flottant
    mat2.setInitFloat(m1, 2, 2);
    // initialisation par copie du tableau de int
    mat3.setInitInt(m2, 2, 2);
    // initialisation de la matrice identite de taille 4
    mat4.setIdentite(4);
    // initialisation d'un vecteur de dimension (n,1) dont les cases sont
    initialisees a 1
    mat5.setOneVector(4);

    //
}
```

Cette bibliothèque propose ensuite les fonctionnalités suivantes sur les objets de type MatriceFloat, avec les méthodes suivantes :

- *float getCase(int i, int j)*, renvoie l'élément à la ligne i et à la colonne j de la matrice. Produit une erreur "index out of range" si les indices ne sont pas valides.

- *boolean setCase(float f)*, modifie une case de la matrice par la valeur f, et renvoi false si échoue.

- *void affichage()*, affiche la matrice.

- *MatriceFloat sumMat(MatriceFloat m)*, renvoie la somme des deux matrices.

- *MatriceFloat prodMat(MatriceFloat m)*, renvoie le produit matriciel des deux matrices .

- *MatriceFloat multScalaire(float scalaire)*, renvoie le produit d'un scalaire et de la matrice.

- *MatriceFloat transpose(MatriceFloat m)*, renvoie la transposée de la matrice.

- *MatriceFloat inverse()*, renvoie l'inverse de la matrice et null si la matrice n'est pas inversible.

- *float determinant()*, renvoie le déterminant de la matrice.

- *float normeVect()*, renvoie la norme euclidienne d'un vecteur colonne. Renvoie 0 si ce n'est pas un vecteur colonne ou si c'est le vecteur colonne nul.

- *float Puissancevpvectp()*, renvoie le rayon spectral de la matrice par la méthodes des puissances si celui ci contient des valeurs propres réelles. la fonction affiche aussi le vecteur propre associé.

La précision est par défaut de 1.0E-5.

- *float PuissancevpvectpWithAcc(float seuil)*, identique à Puissancevpvectp() en valeur de retour avec une précision de seuil.

- *float PuissanceInverse(float nu)*, affiche le vecteur propre associé et renvoi la valeur propre la plus proche de nu. La précision est par défaut de 1.0E-5.

- *float PuissanceInverseWithAcc(float nu, float seuil)*, identique à PuissanceInverse() en valeur de retour avec une précision de seuil.

- *MatriceFloat algorithmeQR(int itération)*, renvoie une matrice triangulaire dont les valeurs propres sont situés sur la diagonale. Le nombre d'itération est limité et peut poser des débordements de tas si sa valeur est trop élevée

4 Limitations des extensions

Les extensions ayant été faites en fin de projet il a fallu faire des choix de conception, et procéder à certaines concessions. Nous allons donc décrire ici les limitations de notre conception.

Tout d'abord, au niveau mémoire, les tableaux sont très gourmands. En effet, nous allouons sur le tas une place égale au nombre d'élément plus un. Ces adresses mémoire doivent de plus être consécutives. Ainsi une matrice 10x10 nécessite un bloc de 101 adresses sur le tas. De plus ces tableaux ne peuvent jamais être désalloués au cours d'un programme.

Notre bibliothèque de calcul matriciel propose des outils très utiles, mais eux aussi limités, notamment en terme de précision et de mémoire. Bien que nous ayons laissé parfois la possibilité à l'utilisateur de régler lui même son seuil d'erreur, nous avons un seuil par défaut de 10^{-5}. De plus les méthodes algorithmeQR, PuissanceInverse et Puissancevectvp peut provoquer des débordements de tas car l'allocation de tableau n'est jamais faite. Les méthodes de PuissanceInverse et Puissancevectvp peuvent aussi ne pas fonctionner si les valeurs propres sont complexes. Il faut donc avoir un regard critique sur les résultats de ces méthodes.