*Ressource 01 – Description du langage **ImpL***

Vous trouverez dans ce document la description complète du langage que nous utiliserons en TDs pour créer un premier compilateur. Ce langage, volontairement simple, vous permettra tout de même de voir la plupart des concepts d'un compilateur. Le projet sera là pour vous faire implémenter les autres.

Général

De la même manière que le langage C, les points virgules sont obligatoires à chaque fin d'expression dans notre langage.

Types

Notre langage admet 4 types de données différents que voici :

- int : nombre entier relatif,
- float : nombre décimal,
- bool : un booléen pouvant prendre deux valeurs : YES et NO,
- array of T : un tableau de type T (T étant un des 4 types autorisés).

Le seul changement de type autorisé de manière implicite étant celui permettant de transformer un int en float (et non l'inverse !).

Valeurs

Afin de pouvoir utiliser nos différents types de données, nous définissons les valeurs que peut accepter notre langage :

- un nombre entier relatif composé d'au moins un chiffre et pouvant être précédé du signe « moins »,
- un nombre décimal composé d'au moins un chiffre, un point et un chiffre pouvant être précédé du signe « moins »,
- les littéraux YES et NO qui sont les valeurs qu'un booléen peut prendre.

Variables

Comme tout bon langage, il nous est nécessaire d'avoir la possibilité de déclarer des variables et de les utiliser. De manière à ce que chaque variable soit identifiée, nous définissons un identifiant de variable comme étant une suite de caractère alphanumérique et d'underscore qui commence par une lettre minuscule.

La déclaration de variable se fait de la manière suivante : <type> <identifiant>.

Afin d'accéder à une variable, on utilise uniquement son identifiant : <identifiant>.

Les variables sont fortement typées et donc ne peuvent pas changer de type au cours de l'exécution du programme.

Affectation de valeur à une variable

L'affectation de valeur à une variable peut prendre deux formes suivant si celle-ci est un tableau ou non.

Affectation d'une valeur à une variable « normal » : <identifiant> = <valeur>. Le type de la valeur doit bien entendu correspondre au type de la variable utilisée. (On ne met pas un float dans un int mais l'inverse est possible !)

Affectation d'une valeur à un tableau : <identifiant> = new <type of array>[x] où x est un nombre entier ou une variable de type int représentant le nombre de case que doit contenir le tableau.

La suppression mémoire d'un tableau se fera avec l'instruction « delete <identifiant> » où identifiant doit être un tableau. (Cette suppression mémoire n'est pas récursive, dans le cas d'un tableau à deux dimensions, vous devez d'abord supprimer la dimension interne)

Affectation d'une valeur à une case d'un tableau : <identifiant>[index_case] = <valeur>.

Expression mathématique

Ce langage donne accès à plusieurs opérateurs mathématiques « standards » qui peuvent être appliqués sur des int ou des float. Les opérateurs sont les suivants :

- « + », l'addition de deux nombres,
- « - », la soustraction de deux nombres ou la multiplication par -1 du nombre à droite (opération unaire),
- « * », la multiplication de deux nombres,
- « ^ », la mise en puissance d'un nombre par un autre (3^2 = 3^2),
- « / », la division de deux nombres,
- « % », le modulo d'un nombre par un autre.

Les 5 premiers peuvent s'appliquer indépendamment sur des int ou des float tandis que le dernier ne peut être appliqué que sur des int.

Hormis dans le dernier cas, les deux types de nombres peuvent être mixé dans ces expressions, on définit le type de l'expression comme étant :

- Un int si les deux composantes sont des int,
- Un float si au moins une des composantes est un float.

Comparaison et expressions booléenne

Différents opérateurs de comparaison sont définis, ces opérateurs de comparaisons se font sur des int, des float ou des booléens (uniquement les deux derniers). Tous les opérateurs de comparaison retourne des booléens.

- « < », inférieur,
- « <= », inférieur ou égal,
- « >= », supérieur ou égal,
- « > », supérieur,
- « != », différent,
- « == », égal.

Des opérateurs booléens nous permettent de combiner les comparaisons :

- « && » et « and » : un « et » logique,
- « || » et « or » : un « ou » logique,
- « ! » et « not » : un « non » logique (opération unaire),
- « ~ » et « xor » : un « ou exclusif » logique.

Branchement conditionnel

Les mots clés if, elseif et else sont réservés pour les conditions. Une condition permet d'exécuter un bloc d'instruction seulement si une certaine condition est vérifié.

```
if (booléen1) {
	// instruction si booléen1 est vrai
}
elseif (booléen2) {
	// instruction si booléen1 est faux et booléen2 est vrai
}
else {
	// instruction si booléen1 et booléen2 sont faux
}
```

Boucles

Seul deux types de boucles seront disponibles dans ce langage : la boucle while et la boucle for.

```
while (booléen) {
	// exécution du bloc tant que booléen est vrai
}

for(<pré instruction> ; booléen ; <post instruction>) {
	// exécution du bloc tant que booléen est vrai
}
```

Fonctions

Dernier élément de notre langage, les fonctions sont à la base de tout programme. Une fonction sera déclarée sous la forme suivante :

```
<type_retour> <identifiant>(<type parametre> <identifiant>, …) {
	//Instruction dans la fonction

	return <valeur/identifiant>
}
```

Le type de retour peut être soit un des types de données autorisés, soit « void » pour déclarer une procédure.

La fonction qui sera automatiquement lancée lors du démarrage de votre programme devra être nommée « main ». (Vous n'avez pas à gérer les paramètres de la ligne de commande)