# Un interpréteur du langage asd2

Projet encadré sur les 3 dernières séances de TP

**Consignes :**

- Ce projet doit être réalisé en binômes.
- Un fichier archive au format `zip` ou `tar.gz` contenant le rapport au format `pdf` et les programmes `C++` sera rendu au plus tard le 26 avril 2019 à 18h.
- Ce fichier sera déposé sur `madoc` par un et un seul membre du binôme dans l'espace devoir correspondant à son groupe de TP.

## 1 Contexte

### 1.1 Le langage asd2

Le langage asd2 est un langage très simple comportant des blocs délimités par des accolades et possédant deux instructions. La première instruction consiste à déclarer et initialiser une variable entière : *var val* déclare une variable *var* de valeur *val*. La seconde instruction consiste à afficher la valeur d'une variable entière à l'écran : *> var* affiche la valeur de la variable *var*. Le nom d'une variable est une séquence de lettres.

Un exemple de programme valide est donné ci-dessous. Un programme possède un bloc principal. Dans chaque bloc, on peut déclarer des variables, afficher des variables ou définir d'autres blocs. Dans un même bloc, on ne peut pas définir deux variables ayant le même nom. L'affichage d'une variable dans un bloc nécessite de retrouver la variable dans ce bloc ou dans le premier bloc en remontant dans l'imbrication des blocs.

```
1  {
2      x 3
3      y 5
4      {
5          z 8
6          {
7              > x                    affiche x = 3
8              y 6
9              > y                    affiche y = 6
10         }
11         > y                        affiche y = 5
12         {
13             z 4
14             > z                    affiche z = 4
15             > x                    affiche x = 3
16         }
17         p 10
18         > z                        affiche z = 8
19         > x                        affiche x = 3
20     }
21     > y                            affiche y = 5
22 }
```

### 1.2 Un interpréteur

Un interpréteur du langage asd2 consiste à saisir un programme ligne par ligne en exécutant chaque instruction à la volée (sans attendre la fin du programme). Ainsi, chaque ligne doit être lue et analysée. Une classe `parser` est fournie pour réaliser cette analyse syntaxique. Puis, en fonction du contenu de la ligne, on réalise l'action qui permet une exécution correcte du programme, en particulier pour afficher les valeurs des variables.

## 2 Travail

L'exécution d'un programme nécessite de maintenir une structure de blocs. Le travail consiste à définir cette structure sous la forme SDA / SDC et à implémenter l'interpréteur en `C++`. Le rapport contiendra uniquement les réponses aux questions 1 à 6.

1. Définir une SDA représentant un bloc, un bloc contenant simplement une collection de variables. Chaque opération sera définie avec sa signature, son rôle et les pré-conditions éventuelles.
2. Proposer et décrire une SDC pour coder un bloc permettant un accès à la valeur d'une variable en $\Theta(1)$ en moyenne.
3. Définir une SDA représentant la structure des blocs maintenue pendant l'exécution d'un programme. Cette structure doit permettre de retrouver les valeurs des variables pour les afficher. Chaque opération sera définie avec sa signature, son rôle et les pré-conditions éventuelles.
4. Proposer et décrire une SDC pour coder la structure des blocs. Donner les ordres de complexité des opérations.
5. Supposons que lors de la fermeture d'un bloc on souhaite afficher la liste des variables déclarées dans ce bloc dans leur ordre d'apparition. Que faut-il comme structure de données pour réaliser cette action ?
6. Citer les différentes erreurs pouvant se produire lors de l'exécution d'un programme.

Enfin, il faut développer cet interpréteur en `C++`. Lors d'une erreur détectée lors de l'interprétation, cette erreur est signalée à l'utilisateur par un simple affichage et l'interprétation continue. Les éléments suivants seront appréciés lors de l'évaluation du code :

— l'organisation du code sous forme de classes ;
— l'indentation automatique lors de l'interprétation d'un programme (l'utilisateur n'a pas à taper les espaces en début de ligne et les déclarations dans un bloc sont alignées) ;
— les commentaires dans le code (consulter la classe `parser` pour comprendre ce qui est demandé).

Des questions subsidiaires sont données ci-dessous sans obligation de les traiter. Il en sera fait mention dans le rapport et cela pourra rapporter des points de bonus.

7. Augmenter le langage asd2 avec de nouvelles instructions, par exemple une incrémentation de la valeur d'une variable de la forme $\underline{+\ var}$.
8. Afficher la liste des variables déclarées dans un bloc lors de la fermeture de celui-ci, dans leur ordre d'apparition.
9. Afficher les numéros de ligne d'un programme.