TD/ TP : Analyse sémantique

Soit le programme suivant écrit dans un langage, auquel nous chercherons à créer un analyseur syntaxique.

Code 1

```
{
      var variable;
      var a,b,c;
      a=4 ;
}
```

A. Analyse lexicale et syntaxique

Reprendre le code de l'analyse lexical et syntaxique créer dans le TP précèdent et l'adapter pour ce langage

B. Analyse sémantique

Dans ce TP nous allons utiliser implémenter le dictionnaire par un tableau à accès séquentielle.

1. Déclarer un tableau appelé **vars** qui va contenir les identifiants détectés dans le code source (code 1).

Soit le tableau **char tokens [8][8]** qui va contenir la liste des tokens du langage

2. Créer une méthode **void createIdentif(char *s,char t[][20])** qui permet d'ajouter l'identifiant au dictionnaire à la suite d'une instruction de déclaration.
3. Créer une fonction **int isToken(char *s,char t[][8])** qui vérifier si le programmeur a déclaré un token comme identifiant.
4. Créer une fonction **int isDeclared(char *s,char t[][20])** qui permet de vérifier si l'identifiant est déjà déclaré.
5. Créer une fonction **void isUsable(char *s,char t[][20])** qui permet de vérifier si le programmeur utilise une variable non déclarée.
6. Créer une fonction **int identifTest(char *s,char t[][8],char vars[][20])** qui appel les fonctions définit dans les **questions 3 et 4** pour vérifier la validité de l'identifiant avant de l'ajouter au dictionnaire (appel de la méthode createIdentif). La fonction identifTest permet de retourner les messages suivants :
   - Déclaration d'une variable avec un nom de token.
   - Déclaration de deux variables avec le même nom

   Utiliser la fonction yyerror() pour afficher les messages d'erreur.
7. Générer le compilateur et tester le avec différents code source.