DEUG MIAS2. Institut Galilée
Option : Langages formels et compilation. 1999–2000
TP 3.

N'oubliez pas de garder une trace de votre travail ...

Dans le TP précédent, on introduisait, dans le programme LEX lui–même des listes de mots à chacune desquelles était associée une classe grammaticale. Le présent sujet remplace la donnée *a priori* de telles listes par celle d'une *table de symboles* : une partie du programme permet de ranger des mots dans cette table, ainsi que leur classe grammaticale, l'autre de retrouver la classe grammaticale des mots d'une phrase. Voici un exemple d'exécution du programme que l'on se propose d'obtenir (Les lignes commençant par `>` sont le résultat de l'exécution, les autres sont entrées au clavier. Pour simplifier, on n'utilise que des minuscules non accentuées.) :

```
verbe es ai prends mange
nom pain pomme main table
je prends une pomme sur la table
> je : inconnu
> prends : verbe
> une : inconnu
> pomme : nom
> sur : inconnu
> la : inconnu
> table : nom
pronom : je tu il, le la les
```

(suite)

```
article un une, le la les
je prends le pain et je le mange
> je : pronom
> prends : verbe
> le : article pronom
> pain : nom
> et : inconnu
> je : pronom
> le : article pronom
> mange : verbe
```

Et ainsi de suite ... Il faut prévoir une sortie honorable.

- Le nom d'une classe grammaticale (`verbe`, `nom`, `pronom`, ...), **placé en début de ligne**, provoque le rangement dans la table des mots qui suivent (chaque mot ainsi rangé doit être accompagné de la classe qui vient de lui être attribuée).
- Dans les autres cas, chaque mot d'une ligne entrée au clavier sera recherché dans la table (telle qu'elle se présente à ce moment) et sera affiché à l'écran avec le résultat de cette recherche.

Voici quelques indications qui vous seront utiles pour écrire un programme LEX réalisant ces opérations :

```
Type
   Classes = ( recherche,                (* classe ``à déterminer'' *)
               article, nom, pronom, verbe, adjectif,
               conjonction,            (* et, mais, ou, car, donc, ...*)
               preposition                     (* a (à), en, de, ...*)
               ) ;
```

```
      MotCl = record
                    mot : string ;                          (* le mot *)
                 classe : Classes                         (* sa classe *)
              end ;

Var
       mode : Classes ;
      table : array[1..Max] of MotCl ;
     ltable : integer ;     (* longueur effective de la table <= Max *)

procedure ranger( c : Classes ; m : string ) ;  (* tente de ranger le
     mot m de classe c dans la table. Ce mot peut être :
     - nouveau,
     - déjà dans la table mais dans une autre classe,
     - déjà dans la table et dans la même classe.
     Dans ce dernier cas, la table n'est pas modifiée, sinon, il faut
     effectivement y ranger le couple c, m. *)

procedure rechercher( m : string ) ;  (* recherche le mot m dans la
     table, l'affiche ainsi que les classes auxquelles il appartient
     ou la mention inconnu lorsque m ne figure pas dans la table. *)


%{
           (* déclarations Turbo Pascal utilisées dans yylex *)
%}
...
%%
...                                          (* prevoir une sortie ...*)
^article  mode := article ;
...
[a-z]+   if mode = recherche then rechercher( yytext )
         else ranger( mode, yytext ) ;
...
\n       mode := recherche ;
.        writeln( yytext , ' n''est pas valide.' ) ;
%%
begin
           mode := recherche ;
           yylex
end.
```

Exercice 1.

Ecrire, compiler et tester un programme obtenu par LEX sur le modèle précédent, en définissant la table des symboles comme un tableau.

Exercice 2. *(pour les amateurs de pointeurs)*

Idem, en définissant la table comme une liste chaînée (`ranger` dans l'ordre alphabétique).