Compilation : deuxième TP
Un mini compilateur Pascal vers C

Maîtrise, 1$^{\text{er}}$ semestre 2001/2002

Dans ce deuxième TP, on se propose d'écrire un compilateur Pascal vers C, pour un fragment très simple de Pascal. À cet effet, nous vous fournissons un squelette de ce compilateur, qu'il vous faudra compléter. Pour le récupérer, exécutez la commande suivante :

```
tar zxf ~filliatr/lib/p2c.tar.gz
```

qui aura pour effet de créer localement un répertoire `p2c` contenant les fichiers suivants :

`pascal.mli`	fichier contenant les types définissant la syntaxe abstraite de Pascal
`lexer.mli, lexer.mll` ...	analyse lexicale
`parser.mly`	analyse syntaxique
`compile.mli, compile.ml`	production du code C
`main.ml`	programme principal
`Makefile`	fichier contenant les règles permettant d'automatiser la compilation (la commande `make` fabrique un exécutable `p2c.opt` à partir des fichiers sources)
`.depend`	fichier contenant les dépendances entre les différents modules Caml (la commande `make depend` le met à jour)

1 Syntaxe (simplifiée) du Pascal

On considère ici un fragment très simple de Pascal, dont la syntaxe concrète est décrite ci-dessous.

Conventions lexicales. Les identificateurs contiennent uniquement des caractères alphanumériques et commencent nécessairement par une lettre. Tous les identificateurs, y compris les mots clés, sont insensibles à la casse (i.e. majuscules et minuscules sont identifiées). Une constante entière est une suite non vide de chiffres de 0 à 9.

Les commentaires sont ouverts par '`(*`' ou '`{`' et fermés par '`*)`' ou '`}`'. Ils ne peuvent être imbriqués.

Grammaire. La grammaire est donnée figure 1. Tous les opérateurs associatifs sont considérés comme associant à gauche. Les précédences sont les plus fortes pour les opérateurs arithmétiques (avec entre eux les précédences usuelles), puis pour ceux de comparaisons, et enfin pour les connectives logiques.

2 Traduction vers le C

On fera en premier lieu l'hypothèse que le fichier Pascal à traduire est syntaxiquement et sémantiquement correct. D'autre part, on ne traduira qu'une unique primitive, `writeln`, dont on supposera qu'elle est toujours utilisée avec un unique argument de type entier.

3 Travail demandé

Complétez les fichiers qui vous sont fournis pour obtenir un compilateur pour le fragment de Pascal considéré ici. Pour information, les fichiers fournis implantent toute la syntaxe donnée sauf les expressions arithmétiques et les appels de fonctions et de procédures.

S'il vous reste du temps, ajoutez les flottants (type `real`) et les tableaux.

```
  program  ::= PROGRAM ident ; variables procfuns BEGIN instrs END .
variables  ::= ϵ
             | VAR vardecls ;
 vardecls  ::= vardecl
             | vardecls ; vardecl
  vardecl  ::= idents : type
   idents  ::= ident
             | ident , idents
     type  ::= INTEGER
             | BOOL
 procfuns  ::= ϵ
             | procfun ; procfuns
  procfun  ::= PROCEDURE ident ( vardecls ) ; variables BEGIN instrs END
             | FUNCTION ident ( vardecls ) : type ; variables BEGIN instrs END
    block  ::= instr
             | BEGIN instrs END
   instrs  ::= instr ;
             | instr ; instrs
    instr  ::= ident := expr
             | ident ( exprs )
             | IF expr THEN block ELSE block
             | IF expr THEN block
     expr  ::= entier
             | TRUE
             | FALSE
             | ident
             | ident ( exprs )
             | expr binop expr
             | unop expr
             | ( expr )
    exprs  ::= ϵ
             | expr
             | expr , exprs
    binop  ::= AND | OR | = | <> | < | <= | > | >= | + | - | * | / | mod
     unop  ::= - | NOT
```

FIG. 1 – Grammaire (simplifiée) du Pascal