IFIPS – Cycle Ingénieur 2e année Informatique
Projet de compilation 2006–2007

Compilateur J--

première partie : analyse syntaxique

Dans la suite, nous utilisons les notations suivantes dans les grammaires :

$\langle règle\rangle^\star$	répétition de la règle $\langle règle\rangle$ un nombre quelconque de fois (y compris aucune)
$\langle règle\rangle_t^\star$	répétition de la règle $\langle règle\rangle$ un nombre quelconque de fois (y compris aucune), les occurrences étant séparées par le terminal t
$\langle règle\rangle^+$	répétition de la règle $\langle règle\rangle$ au moins une fois
$\langle règle\rangle_t^+$	répétition de la règle $\langle règle\rangle$ au moins une fois, les occurrences étant séparées par le terminal t
$\langle règle\rangle$?	utilisation optionnelle de la règle $\langle règle\rangle$ (i.e. 0 ou 1 fois)
($\langle règle\rangle$)	parenthésage ; attention à ne pas confondre ces parenthèses avec les terminaux `(` et `)`

1 Analyse lexicale

Espaces, tabulations et retour-chariots constituent les blancs. Les commentaires débutent par `(*` et s'étendent jusqu'à `*)`, et ils ne doivent pas être imbriqués. Les identificateurs obéissent à l'expression régulière $\langle ident\rangle$ suivante :

$$\begin{array}{lcl}
\langle chiffre\rangle & ::= & \mathtt{0}\text{–}\mathtt{9} \\
\langle alpha\rangle & ::= & \mathtt{a}\text{–}\mathtt{z} \mid \mathtt{A}\text{–}\mathtt{Z} \\
\langle ident\rangle & ::= & \langle alpha\rangle\ (\langle alpha\rangle \mid _ \mid \langle chiffre\rangle)^\star
\end{array}$$

Les identificateurs suivants sont des mots clés :

```
and     begin  bool  class  do        else
end     false  if    in     inherits  int
let     new    nop   not    null      or
string  then   this  true   unit      while
```

Les constantes entières obéissent à l'expression régulière $\langle entier\rangle$ suivante :

$$\langle entier\rangle \quad ::= \quad \langle chiffre\rangle^+$$

Les chaînes de caractères sont délimitées par le caractère `"`. Elles peuvent contenir n'importe quels caractères, à l'exception de `"`, `\` et du retour-chariot. Ces trois caractères doivent être encodés (à l'intérieur d'une chaîne) par les séquences `\"`, `\\` et `\n`, respectivement.

$$
\begin{array}{lll}
\langle \mathit{fichier} \rangle & ::= & \langle \mathit{classe} \rangle^{\star}\ \langle \mathit{expr} \rangle\ \texttt{EOF} \\
\langle \mathit{classe} \rangle & := & \texttt{class}\ \langle \mathit{ident} \rangle\ (\texttt{inherits}\ \langle \mathit{ident} \rangle)\,?\ \texttt{begin}\ (\langle \mathit{atrb} \rangle \mid \langle \mathit{constr} \rangle \mid \langle \mathit{meth} \rangle)^{\star}\texttt{end} \\
\langle \mathit{meth} \rangle & := & \langle \mathit{type} \rangle\ \langle \mathit{ident} \rangle\texttt{(}\ (\langle \mathit{type} \rangle\ \langle \mathit{ident} \rangle)^{\star}_{;}\ \texttt{)}\ \langle \mathit{expr} \rangle \\
\langle \mathit{constr} \rangle & := & \langle \mathit{ident} \rangle\texttt{(}\ (\langle \mathit{type} \rangle\ \langle \mathit{ident} \rangle)^{\star}_{;}\ \texttt{)}\ \langle \mathit{expr} \rangle \\
\langle \mathit{atrb} \rangle & := & \langle \mathit{type} \rangle\ \langle \mathit{ident} \rangle^{+}_{,}\ \texttt{;} \\
\langle \mathit{expr} \rangle & := & \texttt{nop} \mid \langle \mathit{lvalue} \rangle \mid \texttt{this} \mid \langle \mathit{const} \rangle \mid \langle \mathit{expr} \rangle\ \langle \mathit{op} \rangle\ \langle \mathit{expr} \rangle \mid \langle \mathit{unop} \rangle\ \langle \mathit{expr} \rangle \mid \\
 & & \texttt{new}\ \langle \mathit{ident} \rangle\texttt{(}\ \langle \mathit{expr} \rangle^{\star}_{,}\texttt{)} \mid \langle \mathit{lvalue} \rangle\ \texttt{:=}\ \langle \mathit{expr} \rangle \mid \langle \mathit{expr} \rangle\texttt{.}\langle \mathit{ident} \rangle\texttt{(}\ \langle \mathit{expr} \rangle^{\star}_{,}\ \texttt{)} \mid \\
 & & \texttt{let}\ \langle \mathit{ident} \rangle\ \texttt{=}\ \langle \mathit{expr} \rangle\ \texttt{in}\ \langle \mathit{expr} \rangle \mid \langle \mathit{ident} \rangle\texttt{(}\ \langle \mathit{expr} \rangle^{\star}_{,}\ \texttt{)} \mid \langle \mathit{expr} \rangle\ \texttt{;}\langle \mathit{expr} \rangle \mid \\
 & & \texttt{while}\ \langle \mathit{expr} \rangle\ \texttt{do}\ \langle \mathit{expr} \rangle \mid \texttt{if}\ \langle \mathit{expr} \rangle\ \texttt{then}\ \langle \mathit{expr} \rangle\ (\texttt{else}\ \langle \mathit{expr} \rangle)\,? \mid \\
 & & \texttt{begin}\ \langle \mathit{expr} \rangle\ \texttt{end} \mid \texttt{(}\ \langle \mathit{expr} \rangle\ \texttt{)} \\
\langle \mathit{lvalue} \rangle & := & \langle \mathit{ident} \rangle \mid \langle \mathit{expr} \rangle\ \texttt{.}\ \langle \mathit{ident} \rangle \\
\langle \mathit{op} \rangle & := & \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{/} \mid \texttt{<=} \mid \texttt{>=} \mid \texttt{>} \mid \texttt{<} \mid \texttt{<>} \mid \texttt{=} \mid \texttt{and} \mid \texttt{or} \\
\langle \mathit{unop} \rangle & := & \texttt{-} \mid \texttt{not} \\
\langle \mathit{const} \rangle & ::= & \texttt{true} \mid \texttt{false} \mid \langle \mathit{entier} \rangle \mid \langle \mathit{chaîne} \rangle \mid \texttt{null} \\
\langle \mathit{type} \rangle & ::= & \texttt{bool} \mid \texttt{int} \mid \texttt{string} \mid \texttt{unit} \mid \langle \mathit{ident} \rangle
\end{array}
$$

FIG. 1 – Grammaire des fichiers de J--

2 Analyse syntaxique

La grammaire des fichiers sources considérée est donnée figure 1. Le point d'entrée est le non-terminal $\langle \mathit{fichier} \rangle$.

Les associativités et précédences des divers opérateurs ou construction sont données par la table suivante, de la plus faible à la plus forte précédence :

opérateur ou construction	**associativité**
`let in`	—
`;`	à gauche
`while`	—
`if then else`	—
`:=`	à droite
`or`	à gauche
`and`	à gauche
`<   <=   >   >=   =   <>`	à gauche
`+   -`	à gauche
`*   /`	à gauche
`-` (unaire)	—
`not`	—
`.`	à gauche

3 Localisation des erreurs

Lorsqu'une erreur lexicale ou syntaxique est détectée par votre compilateur, elle doit être signalée le plus précisément possible, par sa nature et sa localisation dans le fichier source. On adoptera le format suivant pour cette signalisation :

```
File "test.jmm", line 4, characters 5-6:
syntax error
```

L'anglicisme de la première ligne est nécessaire pour que les outils d'édition de code puissent interpréter la localisation et placer ainsi automatiquement le curseur sur l'emplacement de l'erreur. (En revanche, le message d'erreur proprement dit pourra être écrit en français si vous le souhaitez.)

Anticipation. Dans la deuxième partie, les erreurs sémantiques doivent également être localisées avec précision, c'est-à-dire indiquer le sous-terme en cause. Pour cela il est donc nécessaire que l'arbre de syntaxe abstraite contienne cette information de localisation. Il vous est donc conseillé d'anticiper ce besoin et de construire dès à présent un arbre de syntaxe abstraite adéquat.

4 Travail demandé

Écrire un compilateur, appelons-le `compilo`, acceptant sur sa ligne de commande exactement un fichier `J--` (portant l'extension `.jmm`) et éventuellement l'option `-parse-only`. Cette option est sans effet dans cette première partie et son utilisation sera justifiée par la suite.

Si le fichier est conforme à la syntaxe décrite dans ce document, le programme doit terminer avec le code de sortie 0 (terminaison normale du programme), sans rien afficher. En cas d'erreur lexicale ou syntaxique, celle-ci doit être signalée (de la manière indiquée ci-dessus) et le programme doit terminer avec le code de sortie 1. En cas d'autre erreur (une erreur du compilateur lui-même), le programme doit terminer avec le code de sortie 2.

Votre code devra comprendre au moins les fichiers suivants :
- `lexer.flex` : le lexeur Jflex ;
- `parser.cup` : le parseur CUP ;
- `main.java` : le fichier principal.

Il est possible d'utiliser d'autres fichiers, si cela est justifié dans le rapport.

Modalités de remise de votre projet. Votre projet doit se présenter sous forme d'une archive contenant les *sources* de votre programme (ne donnez pas les fichiers compilés) ainsi que votre rapport. Le projet est à faire impérativement en binôme. Il doit être envoyé par e-mail à `Jean-Christophe.Filliatre@lri.fr`.