

The Language shl

BNF-converter

April 13, 2015

This document was automatically generated by the *BNF-Converter*. It was generated together with the lexer, the parser, and the abstract syntax module, which guarantees that the document matches with the implementation of the language (provided no hand-hacking has taken place).

The lexical structure of shl

Identifiers

Identifiers $\langle Ident \rangle$ are unquoted strings beginning with a letter, followed by any combination of letters, digits, and the characters `_` `'`, reserved words excluded.

Literals

Integer literals $\langle Int \rangle$ are nonempty sequences of digits.

String literals $\langle String \rangle$ have the form `"x"`, where *x* is any sequence of any characters except `"` unless preceded by `\`.

Reserved words and symbols

The set of reserved words is the set of terminals appearing in the grammar. Those reserved words that consist of non-letter characters are called symbols, and they are treated in a different way from those that are similar to identifiers. The lexer follows rules familiar from languages like Haskell, C, and Java, including longest match and spacing conventions.

The reserved words used in shl are the following:

| | | |
|----------|---------|----------|
| Boolean | CYA | DO |
| DONE | ELSE | FI |
| FOR | False | IF |
| IN | Integer | PRINT |
| REF | RETURN | RETURNED |
| SOLUTION | String | THEN |
| True | | |

The symbols used in shl are the following:

```

;      =      (
)      ,      ==
!=     <      >
<=     >=     +
-      *      /
++     --

```

Comments

Single-line comments begin with `<==3`.

Multiple-line comments are enclosed with `<=` and `=3`.

The syntactic structure of shl

Non-terminals are enclosed between `<` and `>`. The symbols `::=` (production), `|` (union) and `ε` (empty rule) belong to the BNF notation. All other symbols are terminals.

```

<Prog> ::= SOLUTION <Blk> CYA

<Blk>  ::= <ListDec> <ListStm>

<Stm>  ::= FOR <Ident> IN <Exp> DO <Blk> DONE
          | IF <Exp> THEN <Blk> FI
          | IF <Exp> THEN <Blk> ELSE <Blk> FI
          | RETURN <Exp> ;
          | PRINT <Exp> ;
          | <Exp> ;
          | <Ident> = <Exp> ;

<ListStm> ::= ε
          | <Stm> <ListStm>

```

$$\begin{aligned}
\langle Dec \rangle & ::= \langle Typ \rangle \langle Ident \rangle ; \\
& | \quad \langle Typ \rangle \langle Ident \rangle = \langle Exp \rangle ; \\
& | \quad \langle Typ \rangle \langle Ident \rangle (\langle ListFArg \rangle) \text{ DO } \langle Blk \rangle \text{ RETURNED} \\
\langle FArg \rangle & ::= \langle Typ \rangle \langle Ident \rangle \\
& | \quad \text{REF } \langle Typ \rangle \langle Ident \rangle \\
\langle ListFArg \rangle & ::= \epsilon \\
& | \quad \langle FArg \rangle \\
& | \quad \langle FArg \rangle , \langle ListFArg \rangle \\
\langle ListDec \rangle & ::= \epsilon \\
& | \quad \langle Dec \rangle \langle ListDec \rangle \\
\langle Typ \rangle & ::= \text{Integer} \\
& | \quad \text{Boolean} \\
& | \quad \text{String} \\
\langle Exp \rangle & ::= \langle Exp \rangle == \langle Exp2 \rangle \\
& | \quad \langle Exp \rangle != \langle Exp2 \rangle \\
& | \quad \langle Exp1 \rangle \\
\langle Exp2 \rangle & ::= \langle Exp2 \rangle < \langle Exp3 \rangle \\
& | \quad \langle Exp2 \rangle > \langle Exp3 \rangle \\
& | \quad \langle Exp2 \rangle \leq \langle Exp3 \rangle \\
& | \quad \langle Exp2 \rangle \geq \langle Exp3 \rangle \\
& | \quad \langle Exp3 \rangle \\
\langle Exp3 \rangle & ::= \langle Exp3 \rangle + \langle Exp4 \rangle \\
& | \quad \langle Exp3 \rangle - \langle Exp4 \rangle \\
& | \quad \langle Exp4 \rangle \\
\langle Exp4 \rangle & ::= \langle Exp4 \rangle * \langle Exp5 \rangle \\
& | \quad \langle Exp4 \rangle / \langle Exp5 \rangle \\
& | \quad \langle Exp5 \rangle \\
\langle Exp5 \rangle & ::= \langle Ident \rangle ++ \\
& | \quad \langle Ident \rangle -- \\
& | \quad \langle Ident \rangle (\langle ListIParam \rangle) \\
& | \quad \langle Ident \rangle \\
& | \quad \langle Constraint \rangle \\
& | \quad \langle Exp6 \rangle \\
\langle Exp1 \rangle & ::= \langle Exp2 \rangle \\
\langle Exp6 \rangle & ::= (\langle Exp \rangle)
\end{aligned}$$

$$\begin{aligned}
\langle \textit{Constraint} \rangle &::= \langle \textit{Integer} \rangle \\
&| \langle \textit{BoolT} \rangle \\
&| \langle \textit{String} \rangle \\
\langle \textit{BoolT} \rangle &::= \textbf{True} \\
&| \textbf{False} \\
\langle \textit{IParam} \rangle &::= \langle \textit{Exp} \rangle \\
\langle \textit{ListIParam} \rangle &::= \epsilon \\
&| \langle \textit{IParam} \rangle \\
&| \langle \textit{IParam} \rangle , \langle \textit{ListIParam} \rangle
\end{aligned}$$