

The Language Swifty

BNF-converter

May 9, 2016

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 Swifty

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

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 Swifty are the following:

<code>Array</code>	<code>array</code>	<code>bool</code>
<code>else</code>	<code>false</code>	<code>for</code>
<code>func</code>	<code>if</code>	<code>in</code>
<code>int</code>	<code>of</code>	<code>print</code>
<code>return</code>	<code>struct</code>	<code>true</code>
<code>var</code>	<code>while</code>	

The symbols used in Swifty are the following:

() ->
 = , :
 { } ||
 && == !=
 < > <=
 >= + -
 * / !
 [] .
 &

Comments

Single-line comments begin with `//`.

Multiple-line comments are enclosed with `/*` and `*/`.

The syntactic structure of Swifty

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

$\langle \text{Program} \rangle ::= \langle \text{ListStmt} \rangle$

$\langle \text{Decl} \rangle ::=$
 $\text{func } \langle \text{Ident} \rangle (\langle \text{ListPDecl} \rangle) -> \langle \text{Type} \rangle \langle \text{Stmt} \rangle$
 $|$
 $\text{func } \langle \text{Ident} \rangle (\langle \text{ListPDecl} \rangle) \langle \text{Stmt} \rangle$
 $|$
 $\text{var } \langle \text{Ident} \rangle = \langle \text{Expr} \rangle$
 $|$
 $\text{struct } \langle \text{Ident} \rangle$
 $|$
 $\text{var } \langle \text{Ident} \rangle , \langle \text{ListIdent} \rangle = \langle \text{Expr} \rangle$

$\langle \text{PDecl} \rangle ::= \langle \text{Ident} \rangle : \langle \text{Type} \rangle$

$\langle \text{ListPDecl} \rangle ::=$
 ϵ
 $|$
 $\langle \text{PDecl} \rangle$
 $|$
 $\langle \text{PDecl} \rangle , \langle \text{ListPDecl} \rangle$

$\langle \text{ListStmt} \rangle ::=$
 ϵ
 $|$
 $\langle \text{Stmt} \rangle \langle \text{ListStmt} \rangle$

$\langle \text{ListExpr} \rangle ::=$
 ϵ
 $|$
 $\langle \text{Expr} \rangle$
 $|$
 $\langle \text{Expr} \rangle , \langle \text{ListExpr} \rangle$

$\langle \text{ListAcc} \rangle ::=$
 $\langle \text{Acc} \rangle$
 $|$
 $\langle \text{Acc} \rangle , \langle \text{ListAcc} \rangle$

$$\begin{aligned}
\langle ListIdent \rangle &::= \langle Ident \rangle \\
&| \quad \langle Ident \rangle , \langle ListIdent \rangle \\
\langle ListType \rangle &::= \langle Type \rangle \\
&| \quad \langle Type \rangle , \langle ListType \rangle \\
\langle Block \rangle &::= \{ \langle ListStmt \rangle \} \\
\langle Acc \rangle &::= \langle Ident \rangle \\
&| \quad \langle Acc \rangle \langle ArraySub \rangle \\
&| \quad \langle Acc \rangle \langle StructSub \rangle \\
\langle Stmt \rangle &::= \langle Block \rangle \\
&| \quad \langle Decl \rangle \\
&| \quad \langle Acc \rangle = \langle Expr \rangle \\
&| \quad \langle Acc \rangle , \langle ListAcc \rangle = \langle Tuple \rangle \\
&| \quad \text{while} (\langle Expr \rangle) \langle Stmt \rangle \\
&| \quad \text{for} \langle Ident \rangle \text{ in } \langle Acc \rangle \langle Stmt \rangle \\
&| \quad \text{if} (\langle Expr \rangle) \langle Stmt \rangle \\
&| \quad \text{if} (\langle Expr \rangle) \langle Block \rangle \text{ else } \langle Stmt \rangle \\
&| \quad \text{return} \langle Expr \rangle \\
&| \quad \text{print} \langle Expr \rangle \\
&| \quad \langle FCall \rangle \\
\langle FCall \rangle &::= \langle Ident \rangle (\langle ListExpr \rangle) \\
\langle Expr1 \rangle &::= \langle Expr1 \rangle || \langle Expr2 \rangle \\
&| \quad \langle Expr2 \rangle \\
\langle Expr2 \rangle &::= \langle Expr2 \rangle \&\& \langle Expr3 \rangle \\
&| \quad \langle Expr3 \rangle \\
\langle Expr3 \rangle &::= \langle Expr3 \rangle == \langle Expr4 \rangle \\
&| \quad \langle Expr3 \rangle != \langle Expr4 \rangle \\
&| \quad \langle Expr4 \rangle \\
\langle Expr4 \rangle &::= \langle Expr4 \rangle < \langle Expr5 \rangle \\
&| \quad \langle Expr4 \rangle > \langle Expr5 \rangle \\
&| \quad \langle Expr4 \rangle <= \langle Expr5 \rangle \\
&| \quad \langle Expr4 \rangle >= \langle Expr5 \rangle \\
&| \quad \langle Expr5 \rangle \\
\langle Expr5 \rangle &::= \langle Expr5 \rangle + \langle Expr6 \rangle \\
&| \quad \langle Expr5 \rangle - \langle Expr6 \rangle \\
&| \quad \langle Expr6 \rangle
\end{aligned}$$

$$\begin{aligned}
\langle \text{Expr6} \rangle &::= \langle \text{Expr6} \rangle * \langle \text{Expr7} \rangle \\
&| \langle \text{Expr6} \rangle / \langle \text{Expr7} \rangle \\
&| \langle \text{Expr7} \rangle \\
\langle \text{Expr7} \rangle &::= - \langle \text{Expr8} \rangle \\
&| ! \langle \text{Expr8} \rangle \\
&| \langle \text{Expr8} \rangle \\
\langle \text{Expr8} \rangle &::= \langle \text{Array} \rangle \\
&| \text{array} (\langle \text{Expr} \rangle , \langle \text{Expr} \rangle) \\
&| \langle \text{Tuple} \rangle \\
&| \langle \text{Acc} \rangle \langle \text{ArraySub} \rangle \\
&| \langle \text{Acc} \rangle \langle \text{StructSub} \rangle \\
&| \langle \text{FCall} \rangle \\
&| \langle \text{Constant} \rangle \\
&| \langle \text{Ident} \rangle \\
&| (\langle \text{Expr} \rangle) \\
\langle \text{ArraySub} \rangle &::= [\langle \text{Expr} \rangle] \\
\langle \text{Array} \rangle &::= \{ \langle \text{ListExpr} \rangle \} \\
\langle \text{Tuple} \rangle &::= (\langle \text{Expr} \rangle , \langle \text{ListExpr} \rangle) \\
\langle \text{StructSub} \rangle &::= . \langle \text{Ident} \rangle \\
\langle \text{Expr} \rangle &::= \langle \text{Expr1} \rangle \\
\langle \text{Constant} \rangle &::= \text{false} \\
&| \text{true} \\
&| \langle \text{Integer} \rangle \\
\langle \text{Type} \rangle &::= \text{int} \\
&| \text{bool} \\
&| \text{Array of } \langle \text{Type} \rangle \\
&| (\langle \text{ListType} \rangle) \\
&| \& \langle \text{Type} \rangle
\end{aligned}$$