```
import library.*;

COMPILER PrettyParva $CN
/* Parva level 1.5 grammar for examination - Coco/R for Java
   PrettyPrinter actions
   Java operator precedences
   Supplied Parva Compiler matches this grammar (and has a few extensions)
   P.D. Terry, Rhodes University, 2011 */

  public static boolean
    indented = true;

CHARACTERS
  lf           = CHR(10) .
  backslash    = CHR(92) .
  control      = CHR(0) .. CHR(31) .
  letter       = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz" .
  digit        = "0123456789" .
  nonZeroDigit = "123456789" .
  stringCh     = ANY - '"' - control - backslash .
  charCh       = ANY - "'" - control - backslash .
  printable    = ANY - control .

TOKENS
  identifier   = letter { letter | digit | "_" { "_" } ( letter | digit ) } .
  number       = "0" | nonZeroDigit { digit } .
  stringLit    = '"' { stringCh  | backslash printable } '"' .
  charLit      = "'" ( charCh    | backslash printable ) "'" .

COMMENTS FROM "//" TO lf
COMMENTS FROM "/*" TO "*/"

IGNORE CHR(9) .. CHR(13)

PRODUCTIONS
  PrettyParva                         (. String name; .)
  = "void"                            (. CodeGen.append("void "); .)
    Ident<out name>                   (. CodeGen.append(name); .)
    "(" ")"                           (. CodeGen.append("()"); .)
    Block .

  Block
  = "{"                               (. CodeGen.append(" {"); CodeGen.indent(); .)
      { Statement<!indented> }
    "}"                               (. CodeGen.exdentNewLine();
                                         CodeGen.append("}"); .) .

  Statement<boolean indented>
  =   Block
    | ";"                             (. CodeGen.append(";"); .)
    |                                 (. if (indented) CodeGen.indent();
                                         CodeGen.newLine(); .)
     (   ConstDeclarations
       | VarDeclarations
       | AssignmentStatement
       | IfStatement
       | WhileStatement
       | DoWhileStatement
       | ForStatement
       | BreakStatement
       | HaltStatement
       | ReadStatement
       | ReadLineStatement
       | WriteStatement
       | WriteLineStatement
     )                                (. if (indented) CodeGen.exdent(); .) .

  ConstDeclarations
  = "const"                           (. CodeGen.append("const "); CodeGen.indentNewLine(); .)
    OneConst
    { WEAK ","                        (. CodeGen.append(", "); CodeGen.newLine(); .)
      OneConst
    } WEAK ";"                        (. CodeGen.append(";"); CodeGen.exdent(); .) .

  OneConst                            (. String name; .)
  = Ident<out name>                   (. CodeGen.append(name); .)
    "="                               (. CodeGen.append(" = "); .)
    Constant .

```

```
Constant                          (. String value; .)
=   IntConst<out value>           (. CodeGen.append(value); .)
  | CharConst<out value>          (. CodeGen.append(value); .)
  | "true"                        (. CodeGen.append("true");  .)
  | "false"                       (. CodeGen.append("false"); .)
  | "null"                        (. CodeGen.append("null");  .) .

VarDeclarations
= Type                            (. CodeGen.indentNewLine(); .)
  OneVar
  { WEAK ","                      (. CodeGen.append(", "); .)
    OneVar
  }
  WEAK ";"                        (. CodeGen.append(";"); CodeGen.exdent(); .) .

Type
= BasicType
  [ "[]"                          (. CodeGen.append("[]"); .)
  ] .

BasicType
=   "int"                         (. CodeGen.append("int");  .)
  | "bool"                        (. CodeGen.append("bool"); .)
  | "char"                        (. CodeGen.append("char"); .) .

OneVar                            (. String name; .)
= Ident<out name>                 (. CodeGen.append(name); .)
  [ "="                           (. CodeGen.append(" = "); .)
    Expression
  ] .

AssignmentStatement
= Assignment WEAK ";"             (. CodeGen.append(";"); .) .

Assignment
=   Designator
    (   AssignOp Expression
      | "++"                      (. CodeGen.append("++"); .)
      | "--"                      (. CodeGen.append("--"); .)
    )
  | "++"                          (. CodeGen.append("++"); .)
    Designator
  | "--"                          (. CodeGen.append("--"); .)
    Designator .

Designator                        (. String name; .)
= Ident<out name>                 (. CodeGen.append(name); .)
  [  "["                          (. CodeGen.append("["); .)
        Expression
     "]"                          (. CodeGen.append("]"); .)
  ] .

IfStatement
= "if"                            (. CodeGen.append("if "); .)
  "("                             (. CodeGen.append("("); .)
     Condition
  ")"                             (. CodeGen.append(")"); .)
  Statement<indented>
  { "elsif" "("                   (. CodeGen.newLine(); CodeGen.append("elsif ("); .)
      Condition
    ")"                           (. CodeGen.append(")"); .)
    Statement<indented>
  }
  [ "else"                        (. CodeGen.newLine(); CodeGen.append("else"); .)
    Statement<indented>
  ] .

WhileStatement
= "while"                         (. CodeGen.append("while "); .)
  "("                             (. CodeGen.append("("); .)
     Condition
  ")"                             (. CodeGen.append(")"); .)
  Statement<indented> .

DoWhileStatement
= "do"                            (. CodeGen.append("do"); .)
    Statement<indented>
  WEAK "while"                    (. CodeGen.newLine(); CodeGen.append("while "); .)
  "("                             (. CodeGen.append("("); .)
     Condition
  ")"                             (. CodeGen.append(")"); .)
  WEAK ";"                        (. CodeGen.append(";"); .) .
```

```
163   ForStatement
164   = "for"                           (. CodeGen.append("for "); .)
165     ForControl
166     Statement<indented> .
167
168   ForControl                        (. String name; .)
169   =   "("                           (. CodeGen.append("("); .)
170         [ [ BasicType               (. CodeGen.append(" "); .)
171           ]
172           Ident<out name>           (. CodeGen.append(name); .)
173           "="                       (. CodeGen.append(" = "); .)
174           Expression
175         ] ";"                       (. CodeGen.append("; "); .)
176         [ Condition ] ";"           (. CodeGen.append("; "); .)
177         [ Assignment ]              (. CodeGen.append(")"); .)
178       ")"
179     | Ident<out name>               (. CodeGen.append(name); .)
180       "="                           (. CodeGen.append(" = "); .)
181       Expression
182       (   "to"                      (. CodeGen.append(" to "); .)
183         | "downto"                  (. CodeGen.append(" downto "); .)
184       )
185       Expression .
186
187   BreakStatement
188   = "break"                         (. CodeGen.append("break"); .)
189     WEAK ";"                        (. CodeGen.append(";"); .) .
190
191   HaltStatement
192   = "halt"                          (. CodeGen.append("halt"); .)
193     WEAK ";"                        (. CodeGen.append(";"); .) .
194
195   ReadStatement
196   = "read"                          (. CodeGen.append("read"); .)
197     "("                             (. CodeGen.append("("); .)
198        ReadElement
199        { WEAK ","                   (. CodeGen.append(", "); .)
200          ReadElement
201        }
202     ")"                             (. CodeGen.append(")"); .)
203     WEAK ";"                        (. CodeGen.append(";"); .) .
204
205   ReadLineStatement
206   = "readLine"                      (. CodeGen.append("readLine"); .)
207     "("                             (. CodeGen.append("("); .)
208        [ ReadElement
209          { WEAK ","                 (. CodeGen.append(", "); .)
210            ReadElement
211          }
212        ]
213     ")"                             (. CodeGen.append(")"); .)
214     WEAK ";"                        (. CodeGen.append(";"); .) .
215
216   ReadElement                       (. String str; .)
217   = (   StringConst<out str>        (. CodeGen.append(str); .)
218       | Designator
219     ) .
220
221   WriteStatement
222   = "write"                         (. CodeGen.append("write"); .)
223     "("                             (. CodeGen.append("("); .)
224        WriteElement
225        { WEAK ","                   (. CodeGen.append(", "); .)
226          WriteElement
227        }
228     ")"                             (. CodeGen.append(")"); .)
229     WEAK ";"                        (. CodeGen.append(";"); .) .
230
231   WriteLineStatement
232   = "writeLine"                     (. CodeGen.append("writeLine"); .)
233     "("                             (. CodeGen.append("("); .)
234        [ WriteElement
235          { WEAK ","                 (. CodeGen.append(", "); .)
236            WriteElement
237          }
238        ]
239     ")"                             (. CodeGen.append(")"); .)
240     WEAK ";"                        (. CodeGen.append(";"); .) .
241
242   WriteElement                      (. String str; .)
243   = (   StringConst<out str>        (. CodeGen.append(str); .)
244       | Expression
245     ) .
246
```

```
247   Condition
248   = Expression .
249
250   Expression
251   = AndExp
252     { "||"                          (. CodeGen.append(" || "); .)
253       AndExp
254     } .
255
256   AndExp
257   = EqlExp
258     { "&&"                          (. CodeGen.append(" && "); .)
259       EqlExp
260     } .
261
262   EqlExp
263   = RelExp
264     [ EqualOp
265       RelExp
266     ] .
267
268   RelExp
269   = AddExp
270     [ RelOp
271       AddExp
272     ] .
273
274   AddExp
275   = MultExp
276     { AddOp
277       MultExp
278     } .
279
280   MultExp
281   = Factor
282     { MulOp
283       Factor
284     } .
285
286   Factor
287   =   Primary
288     | "+"                           (. CodeGen.append(" +"); .)
289       Factor
290     | "-"                           (. CodeGen.append(" -"); .)
291       Factor
292     | "!"                           (. CodeGen.append("!"); .)
293       Factor .
294
295   Primary
296   =   Designator
297     | Constant
298     | "new"                         (. CodeGen.append("new "); .)
299       BasicType
300       "["                           (. CodeGen.append("["); .)
301          Expression
302       "]"                           (. CodeGen.append("]"); .)
303     | "("                           (. CodeGen.append("("); .)
304       (   "char" ")"                (. CodeGen.append("char)"); .)
305           Factor
306         | "int" ")"                 (. CodeGen.append("int)"); .)
307           Factor
308         | Expression
309           ")"                       (. CodeGen.append(")"); .)
310       ) .
311
312   AddOp
313   =   "+"                           (. CodeGen.append(" + "); .)
314     | "-"                           (. CodeGen.append(" - "); .) .
315
316   MulOp
317   =   "*"                           (. CodeGen.append(" * "); .)
318     | "/"                           (. CodeGen.append(" / "); .)
319     | "%"                           (. CodeGen.append(" % "); .) .
320
321   EqualOp
322   =   "=="                          (. CodeGen.append(" == "); .)
323     | "!="                          (. CodeGen.append(" != "); .) .
324
325   RelOp
326   =   "<"                           (. CodeGen.append(" < "); .)
327     | "<="                          (. CodeGen.append(" <= "); .)
328     | ">"                           (. CodeGen.append(" > "); .)
329     | ">="                          (. CodeGen.append(" >= "); .) .
330
```

```
331   AssignOp
332   =   "="                           (. CodeGen.append(" = "); .) .
333
334   Ident<out String name>
335   = identifier                      (. name = token.val; .) .
336
337   StringConst<out String name>
338   = stringLit                       (. name = token.val; .) .
339
340   CharConst<out String name>
341   = charLit                         (. name = token.val; .) .
342
343   IntConst<out String name>
344   = number                          (. name = token.val; .) .
345
346 END PrettyParva.
```

```
 1 // Code Generation for Parva level 1 pretty-printers (Java version)
 2 // P.D. Terry, Rhodes University, 2011
 3
 4 package PrettyParva;
 5
 6 import library.*;
 7 import java.util.*;
 8
 9   class CodeGen {
10     public static OutFile output = null;
11
12     static int indentation = 0;
13
14     public static void openOutput(String fileName) {
15     // Opens output file from specified fileName
16       output = new OutFile(fileName);
17       if (output.openError()) {
18         System.out.println("cannot open " + fileName);
19         System.exit(1);
20       }
21     }
22
23     public static void closeOutput() {
24     // Closes output file
25       output.close();
26     }
27
28     public static void append(String str) {
29     // Appends str to output file
30       output.write(str);
31     }
32
33     public static void newLine() {
34     // Writes line mark to output file but leaves indentation as before
35       output.writeLine();
36       appendLeadingSpaces();
37     }
38
39     static void appendLeadingSpaces() {
40     // Appends the spaces needed at the start of each line of output before a statement
41       for (int j = 1; j <= indentation; j++) output.write(' ');
42     }
43
44     public static void indentNewLine() {
45     // Writes line mark to output file and prepares to indent further lines
46     // by two spaces more than before
47       indentation += 2;
48       newLine();
49     }
50
51     public static void exdentNewLine() {
52     // Writes line mark to output file and prepares to indent further lines
53     // by two spaces less
54       indentation -= 2;
55       newLine();
56     }
57
58     public static void indent() {
59     // Increments indentation level by 2
60       indentation += 2;
61     }
62
63     public static void exdent() {
64     // Decrements indentation level by 2
65       indentation -= 2;
66     }
67   } // CodeGen
```

```
COMPILER Mikra $CN
/* Mikra level 1.5 grammar
   P.D. Terry, Rhodes University, 2011 */

CHARACTERS
  lf           = CHR(10) .
  backslash    = CHR(92) .
  control      = CHR(0) .. CHR(31) .
  letter       = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz" .
  digit        = "0123456789" .
  nonZeroDigit = "123456789" .
  stringCh     = ANY - '"' - control - backslash .
  charCh       = ANY - "'" - control - backslash .
  printable    = ANY - control .

TOKENS
  identifier   = letter { letter | digit | "_" { "_" } ( letter | digit ) } .
  number       = "0" | nonZeroDigit { digit } .
  stringLit    = '"' { stringCh  | backslash printable } '"' .
  charLit      = "'" ( charCh    | backslash printable ) "'" .

COMMENTS FROM "(*" TO "*)"

IGNORE CHR(9) .. CHR(13)

PRODUCTIONS
  Mikra
  = "program" Ident ";"
     Block  Ident "." .

  Block
  = { ConstDeclarations | VarDeclarations }
    "begin"  StatementSequence "end" .

  StatementSequence
  = Statement { ";" Statement } .

  Statement
  = SYNC
    [   AssignmentStatement  | IfStatement
      | WhileStatement       | HaltStatement
      | ReadStatement        | WriteStatement
      | ForStatement         | RepeatStatement
      | LoopStatement        | ExitStatement
      | IncOrDecStatement
    ] .

  ConstDeclarations
  = "const" OneConst { "," OneConst } ";" .

  OneConst
  = Ident "=" Constant .

  Constant
  = IntConst | CharConst | "true" | "false" | "null" .

  VarDeclarations
  = "var" VarList ";" { VarList ";" } .

  VarList
   = Ident { "," Ident } ":" Type .

  Type
  = [ "array" "of" ] BasicType .

  BasicType
  = "int" | "bool" | "char" .

  AssignmentStatement
  = Designator AssignOp Expression .

  Designator
  = Ident [ "[" Expression "]" ] .

  IfStatement
  = "if" Condition "then" StatementSequence
     { "elsif" Condition "then" StatementSequence }
     [ "else" StatementSequence ]
    "end" .

  WhileStatement
  = "while" Condition "do" StatementSequence "end" .

```

```
    RepeatStatement
    = "repeat" StatementSequence "until" Condition .

    LoopStatement
    = "loop" StatementSequence "end" .

    ForStatement
    = "for" Ident ":=" Expression ( "to" | "downto" ) Expression "do" StatementSequence "end" .

    HaltStatement
    = "halt" .

    ExitStatement
    = "exit" .

    IncOrDecStatement
    = ( "inc" | "dec" ) "(" Designator ")" .

    ReadStatement
    =   "read"  ReadList
      | "readLn" [ ReadList ] .

    ReadList
    = "(" ReadElement { "," ReadElement } ")" .

    ReadElement
    = StringConst | Designator .

    WriteStatement
    =   "write" WriteList
      | "writeLn" [ WriteList ] .

    WriteList = "(" WriteElement { "," WriteElement } ")" .

    WriteElement
    = StringConst | Expression .

    Condition
    = Expression .

    Expression = AddExp [ RelOp AddExp ] .

    AddExp
    = [ "+" | "-" ] MulExp { AddOp MulExp } .

    MulExp
    = Factor { MulOp Factor } .

    Factor
    =   Designator | Constant
      | "new" BasicType "[" Expression "]"
      | "char" "(" Expression ")"
      | "int"  "(" Expression ")"
      | "(" Expression ")"
      | NotOp Factor .

    NotOp
    = "not" .

    MulOp
    = "*" | "/" | "mod" | "and" .

    AddOp
    = "+" | "-" | "or" .

    RelOp
    = "=" | "<>" | "<" | "<=" | ">" | ">=" .

    AssignOp
    = ":="  .

    Ident = identifier .

    StringConst = stringLit .

    CharConst = charLit .

    IntConst = number .

END Mikra.
```