|  |  |  |  |
| --- | --- | --- | --- |
| File Name | Line(Final) | Why We Changed it | What we Changed it to |
| Parser.pt | 1676 | We fixed the parser by uncommenting it | begin  Initialize;  Parser;  { No sense running semantic phase if serious syntax errors found }  if errorCount > 0 then  rewrite(parseStream {, NparseStream} );  end. { ScannerParser } |
| Parser.ssl | 116-123 | Added sPublic, sDefault, sClass, sLoopStmt, sLoopBreakWhen, sLoopEnd, sSubstring, sLength to Parser.ssl output and removed removed sWhileStmt,sRepeatStmt, sRepeatEnd from Parser.ssl output | sPublic  sDefault  sClass  sLoopStmt  sLoopBreakWhen  sEndLoop  sSubstring  sLength |
| Parser.ssl | 37-46 | * removed pThen, pEnd, pUntil, pDo, pConst, pProdecure, pBegin, pWhile, pRepeat, pColonequals from input Parser.ssl input * added pLet, pFunction, pSwitch, pDefault, pElsif, pLoop, pBreak, pWhen, pClass, pPublic to Parser.ssl input | pLet 'let'  pFunction 'function'  pSwitch 'switch'  pDefault 'default'  pElsif 'elsif'  pLoop 'loop'  pBreak 'break'  pWhen 'when'  pClass 'class'  pPublic 'public; |
| Parser.ssl | 66-80 | * changed pNotEqual from <> to != * added pDoubleEquals, pHash, pAt, pLeftBrace, pRightBrace to synchronize with scan.ssl | pNotEqual '!='  pDoubleEquals '=='  pHash '#'  pAt '@'  pLeftBrace '{'  pRightBrace '}' |
| Parser.def | 64-123 | populated new identifiers for tokens by calling ssl parser.ssl so that token numbers would match | sIdentifier = 0;  firstSemanticToken = 0;  firstCompoundSemanticToken = 0;  sInteger = 1;  sLiteral = 2;  lastCompoundSemanticToken = 2;  sProgram = 3;  sParmBegin = 4;  sParmEnd = 5;  sConst = 6;  sType = 7;  sVar = 8;  sProcedure = 9;  sBegin = 10;  sEnd = 11;  sNegate = 12;  sArray = 13;  sPacked = 14;  sFile = 15;  sRange = 16;  sCaseStmt = 17;  sCaseEnd = 18;  sLabelEnd = 19;  sExpnEnd = 20;  sNullStmt = 21;  sAssignmentStmt = 22;  sSubscript = 23;  sCallStmt = 24;  sFieldWidth = 25;  sIfStmt = 26;  sThen = 27;  sElse = 28;  sPublic = 29;  sDefault = 30;  sClass = 31;  sLoopStmt = 32;  sLoopBreakWhen = 33;  sEndLoop = 34;  sSubstring = 35;  sLength = 36;  sEq = 37;  sNE = 38;  sLT = 39;  sLE = 40;  sGT = 41;  sGE = 42;  sAdd = 43;  sSubtract = 44;  sMultiply = 45;  sDivide = 46;  sModulus = 47;  sInfixOr = 48;  sOr = 49;  sInfixAnd = 50;  sAnd = 51;  sNot = 52;  sNewLine = 53;  lastSemanticToken = 53; |
| Parser.pt | 163-221 | populated new identifiers with values from parser.def | pDiv = 0;  firstKeywordToken = 0;  pMod = 1;  pOr = 2;  pAnd = 3;  pNot = 4;  pElse = 5;  pOf = 6;  pArray = 7;  pFile = 8;  pPacked = 9;  pProgram = 10;  pVar = 11;  pType = 12;  pIf = 13;  pCase = 14;  pLet = 15;  pFunction = 16;  pSwitch = 17;  pDefault = 18;  pElsif = 19;  pLoop = 20;  pBreak = 21;  pWhen = 22;  pClass = 23;  pPublic = 24;  lastKeywordToken = 24;  pIdentifier = 25;  firstCompoundInputToken = 25;  pInteger = 26;  pLiteral = 27;  lastCompoundInputToken = 27;  pNewLine = 28;  pEndFile = 29;  pPlus = 30;  pMinus = 31;  pStar = 32;  pDot = 33;  pComma = 34;  pSemicolon = 35;  pColon = 36;  pEquals = 37;  pNotEqual = 38;  pDoubleEquals = 39;  pLess = 40;  pLessEqual = 41;  pGreaterEqual = 42;  pGreater = 43;  pLeftParen = 44;  pRightParen = 45;  pLeftBracket = 46;  pRightBracket = 47;  pDotDot = 48;  pHash = 49;  pAt = 50;  pLeftBrace = 51;  pRightBrace = 52;  lastSyntaxToken = 52; |
| Parser.pt | 164-222 | added semantic tokens | sIdentifier = 0;  firstSemanticToken = 0;  firstCompoundSemanticToken = 0;  sInteger = 1;  sLiteral = 2;  lastCompoundSemanticToken = 2;  sProgram = 3;  sParmBegin = 4;  sParmEnd = 5;  sConst = 6;  sType = 7;  sVar = 8;  sProcedure = 9;  sBegin = 10;  sEnd = 11;  sNegate = 12;  sArray = 13;  sPacked = 14;  sFile = 15;  sRange = 16;  sCaseStmt = 17;  sCaseEnd = 18;  sLabelEnd = 19;  sExpnEnd = 20;  sNullStmt = 21;  sAssignmentStmt = 22;  sSubscript = 23;  sCallStmt = 24;  sFieldWidth = 25;  sIfStmt = 26;  sThen = 27;  sElse = 28;  sPublic = 29;  sDefault = 30;  sClass = 31;  sLoopStmt = 32;  sLoopBreakWhen = 33;  sEndLoop = 34;  sSubstring = 35;  sLength = 36;  sEq = 37;  sNE = 38;  sLT = 39;  sLE = 40;  sGT = 41;  sGE = 42;  sAdd = 43;  sSubtract = 44;  sMultiply = 45;  sDivide = 46;  sModulus = 47;  sInfixOr = 48;  sOr = 49;  sInfixAnd = 50;  sAnd = 51;  sNot = 52;  sNewLine = 53;  lastSemanticToken = 53; |
| Parser.ssl | 170 | Changed ‘const’ to ‘let’ | | 'let': |
| Parser.ssl | 246/250 | Changed ‘=’ to ‘:’ to alter type declaration syntax | ':' @TypeBody ';' |
| Parser.ssl | 288 | Allowed for multiple variable names in one line |  |
| Parser.ssl | 347 | Changed ‘begin’ to ‘{‘ within the Statement logic | | '{': |
| Parser.ssl | 351 | Changed ‘while’ to ‘loop’ and @WhileStmt to @LoopStmt | | 'loop':  @LoopStmt |
| Parser.ssl | 503 | Added LoopStmt | LoopStmt : |
| Parser.ssl | 179 | Replaced ‘procedure’ with ‘function’ to accommodate change | | 'function': |
| Parser.ssl | 186-187 | Added ‘public’ to emit a public token | | 'public':  'function'  .sProcedure  .sPublic  pIdentifier  .sIdentifier  @ProcedureHeading  '{'  @Block |
| Parser.ssl | 475 | Added ‘{‘ after LoopStmt |  |
| Parser.ssl | 368 | Changed ‘:=’ to ‘=’ | | '=': |
| Parser.ssl | 333 | Replaced semicolon with colon to account for multiple local variables in routines |  |
| Parser.ssl | 185-191 | Added logic to ensure public is followed by function | | 'public':  'function'  .sProcedure  .sPublic  pIdentifier  .sIdentifier  @ProcedureHeading  '{'  @Block |
| Parser.ssl | 195 | Removed { before @BeginStmt as it was consuming an extra character (this allows for the program to emit sBegin and sEnd properly) |  |
| Parser.ssl | 184/192 | Added leftBrace before block to consume input | '{' |
| Parser.ssl | 363 | Removed the repeat option from statement |  |
| Parser.ssl | 380 | Changed := to = for JT syntax | | '=': |
| Parser.ssl | 194 | Added 'class' for appropriate syntax | |'class': |
| Parser.ssl | 450 – 458 | Added logic for case statement | CaseStmt :  .sCaseStmt  '('  pIdentifier  .sIdentifier  .sExpnEnd  ')'  '{'  {[  |'case':  @CaseExpn  .sBegin  {[  |'break':  ';'  >  |\*:  @Statement  ';'  ]}  .sEnd  | \*:  >  ]}  'default'  .sDefault  ';'  @Statement  'break'  ';'  '}'  .sCaseEnd;  CaseExpn:  @OptionallySignedIntegerConstant  ':'; |
| Parser.ssl | 487 - 505 | Added appropriate S/SL logic for Loop statements | LoopStmt :  .sLoopStmt  '{'  .sBegin  {[  |'break':  'when'  .sEnd  .sLoopBreakWhen  @Expression  .sExpnEnd  .sBegin  |'}':  .sEnd  .sEndLoop  >  | \*:  @Statement  ]}; |
| Parser.ssl | 448- 465 | Added logic for ‘if’ statement | IfStmt :  .sIfStmt  '('  @Expression  ')' .sExpnEnd  .sThen  @Statement  [  |pElsif:  .sElse  @IfStmt  |pElse:  .sElse  @Statement  |\*:  %Do nothing  ]; |