/SIMULA PRETTYPRINTER USING PASCAL/

by

JUNG-JUIN CHEN

B.S., FU JEN CATHOLIC UNIVERSITY, 1979

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1986

Approved by:

Major Professor

Acknowledgements

The author wishes to express her sincere gratitude and appreciation to Dr. Rodney M. Bates, for his great guidance and assistance during the course of the work. Thanks are also extended to Dr. Virgil E. Wallentine, for discussing ideas and providing suggestions. The author would also wish to thank Dr. Elizabeth A. Unger and Dr. Austin C. Melton, Jr. for serving as the advisory committee members.

The author wishes to thank her parents Mr. and Mrs. Jih Chen, her parents-in-law Mr. and Mr. Tsung Yung Cheng, and her husband Tai-Ben Cheng for their love and support through the education.

ILLEGIBLE

THE FOLLOWING DOCUMENT (S) IS ILLEGIBLE DUE TO THE PRINTING ON THE ORIGINAL BEING CUT OFF

ILLEGIBLE

TABLE OF CONTENTS

LIST OF FIGURES

page

Figure

Chapter 1

Introduction

1.1 Background

A prettyprinter is a program designed to parse and reformat the whole text of a source program in a desired style for a particular language. Most of the time, programmers need to set up source text according to a standard set of indentation rules by counting the number of times they hit the space bar and make modification by moving a whole section of text to the left or right. A program that has been thoughtfully formatted according to such a standard is said to be prettyprinted. The advantages of a prettyprinter are that it saves programmers a lot of time at the terminal editing source code for both initial text entry and text modification, and it enforces standard indentation conventions so that the nesting can be easily discerned by humans.

1.2 History

Many prettyprinter programs have been written for programming languages(1,2,3,4). There is a Pascal prettyprinter(PPP) written in Pascal by Dr. Rodney M. Bates in May 1981(5). The PPP is now available on the Interdata 8/32 under Unix V7(6) at Kansas State University's Computer Science Department. After the Pascal prettyprinter was done, a lot of

students have received benefits from it. Under Dr. Bates' advising, Wun-Jen Lin, wrote another prettyprinter for the Euclid language as his master's report(7) in 1983. The Euclid prettyprinter(EPP) was implemented by modifying the existing Pascal prettyprinter and using the language called Syntax/Semantic Language(8) instead of Pascal to write the parser.

1.3 Purpose

This project is to implement a prettyprinter program written in Pascal for Simula and make it available on the Interdata 8/32 machine under Unix V7 operating system in the Kansas State University Computer Science Department. Simula, developed by Dahl and Nygaard in Norway(9), is Algol-like in form and has simulation as a major application domain. One important feature which it introduced was the 'class' concept, the idea that a group of declarations and procedures are collected together and treated as a unit. Objects of the class can be generated and they have a life which is independent of the block in which they were created. This is especially useful for applications involving simulation. The class concept is now viewed as the origin of data abstraction.

1.4 Approach

The approach of this project is to use the language called

Syntax Semantic Language(S/SL) to write the parser and modify the existing Euclid prettyprinter written in Pascal. The project can be divided into the following six phases :

1. The description of Simula's syntax accomplished by combining the syntax of Algol 60 (the subset of Simula) and that of the extension part of Simula.
2. The implementation of the Syntax/Semantic Language(S/SL) program for the parser of Simula.
3. The translation of the S/SL program into a S/SL table(or an S-code) via an S/SL assembler.
4. The modification of EPP's lexical scanner to make a new lexical scanner for Simula.
5. The replacement of the S/SL table in the EPP which was modified via step 3 by the S/SL table implemented for Simula to make a new Simula prettyprinter(SPP) program.
6. The implementation of the SPP.

1.5 Organization

This report is organized as follows: Chapter 1 contains explanations of the motivation for designing a prettyprinter, the history of prettyprinter, the purpose of this report, and the approach used to implement this project. Chapter 2 provides the concepts and features of the Syntax/Semantic Language, and the reasons for using it in this project. Chapter 3 contains an explaination of the functions and data structures of the Simula prettyprinter(SPP), and the indentation style used in the Simula prettyprinter. Chapter 4 contains details of the entire process

of building the SPP. Chapter 5 presents the conclusion that summarizes the features of using S/SL in this project, and the nature and results of running the SPP. Appendix 1 is a listing of the SPP program in Pascal. Appendix 2 is a listing of the S/SL program. Appendix 3 includes a listing of the input sample Simula program. Appendix 4 contains a listing of the output sample Simula program.

Chapter 2

Syntax/Semantic Language

2.1 The concepts of S/SL

Syntax/Semantic Language(S/SL) is a programming language developed at the University of Toronto as a tool for implementing compilers. S/SL has been used to construct scanners, parsers, semantic analyzers and code generators.

The complete S/SL includes a subset called SL (Syntax Language) which has the same recognition power as LR(k) parsers, and it can invoke semantic operations implemented in another language such as Pascal. S/SL implies a top down programming methodology. First, a data-free algorithm is developed in S/SL. The algorithm invokes operations on "semantic mechanisms". The semantic mechanism is a abstract object, specified, from the point of view of the S/SL, only by the effect of operations upon the object. Later, the mechanisms are implemented apart from the S/SL.

S/SL assumes a computational model that is illustrated in Figure 1. The major role of an S/SL program is to translate its input stream into an output stream. The input stream consists of tokens. The S/SL program reads(accepts) tokens one-by-one from the input stream and emits tokens to an output stream. It can also emit error signals to an error stream. An S/SL program also supports a set of possibly recursive rules. The implementation uses an implicit stack of return addresses to

allow a called rule to return to the appropriate calling point.
allow a called rule to return to the appropriate calling point.

Besides controlling the input and the output streams, the S/SL program can manipulate data using the semantic mechanisms. The interface to each semantic mechanism is defined in S/SL, but the implementation is done separately in a base language such as Pascal. The S/SL program invokes semantic operations to inspect or manipulate data, but has no access to data such as making an assignment or comparison. This concept of a pure control language and an abstract data (semantic) mechanism imply that the S/SL programmer needs to know only the meaning (specification) of the program without being directly concerned with the implementation of the operations in writing his S/SL program. This is analogous to specifying an abstract data type and using it without being concerned with its implementation.

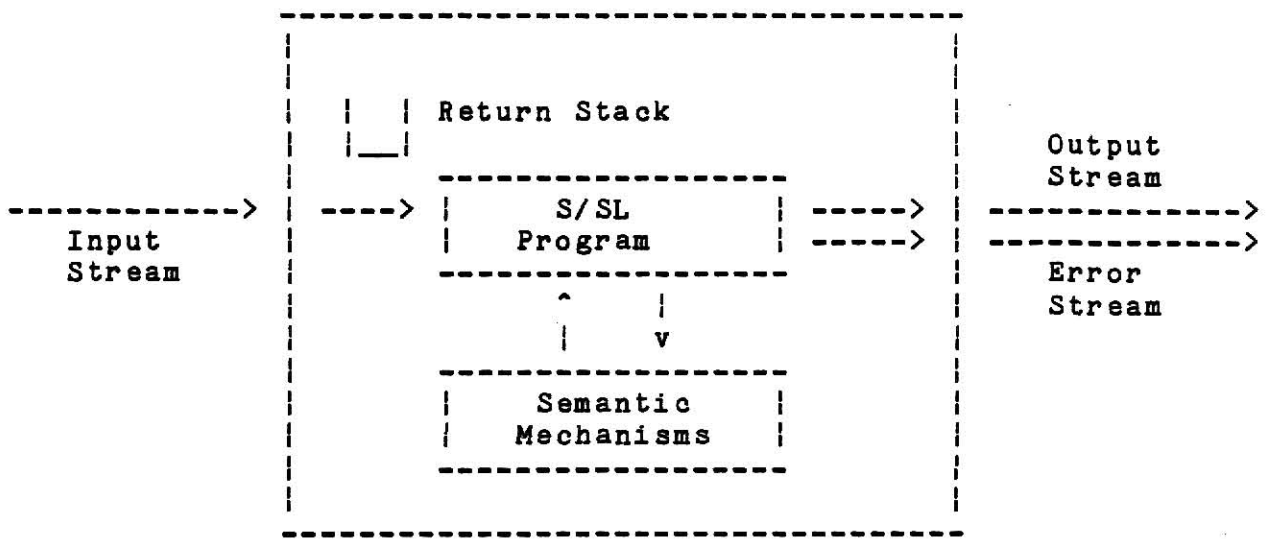

Figure 1. A computational model for S/SL

2.2 The features of S/SL

S/SL includes a subset called SL(Syntax Language) and facilities for invocation of semantic operations which are implemented in a base language such as Pascal. S/SL is a very modest language. Each SL program consists of a list of executable rules. Each rule consists of a name, an optional return type and a sequence of actions (statements). Each rule has one of these two forms:

```
name: actions;

name >> type: actions;
```

a rule with the first form is called a 'procedure rule'; a rule with the second form is called a 'choice rule'. These two forms are analogous to procedures and functions in Pascal. There are eight different actions (statements) in SL. These are described below.

1. The call action. The appearance of @ followed by the name of a procedure rule signifies that the rule is to be called. For example, here is a call to the IfStatement rule:

```
@IfStatement
```

2. The return action. The symbol >> in a rule signifies return before reaching the end of the rule. For example, Here is a choice rule, OptionalExpression, that returns true if the next token is the end-of-file; otherwise it parses an expression and returns false.

```
OptionalExpression >> Boolean:
  [
     | eof:
        >> true
     | *:
```

```
        >> false
    ];
```

3. The input (or match) action. The appearance of an input token in a rule signifies that the next input should be read and must match the token. For example, the following specifies that the next item in the input stream is to be read and it must be a number token:

```
'number'
```

Note that all of the input tokens, output tokens and semantic operations need to be defined in S/SL. This is similar to defining a type in Pascal. For example,

```
inputs:
    number              % input tokens
    plus      '+'
    minus     '-'
    equal     '='
    ... etc ... ;

operations:
    setupPosition       % sets up start position for a token
    blankLine           % handles placement of blank line
    ... etc ... ;
```

Here we have also shown the convention for comments in S/SL programs, namely, anything to the right of a % on a line is ignored.

4. The emit action. The appearance of a dot followed by an output token signifies that the token is to be emitted to a special output stream, for example:

```
.subtract
```

5. The error action. The appearance of # followed by an error signal signifies that the error signal is to be emitted to a special error stream, for example:

```
#missingColon
```

6. The cycle action. Each cycle is of the form:

```
{
   actions
}
```

The enclosed actions are repeated until one of the cycle's exits '>' is encountered.

7. The exit construct. The appearance of '>' signifies exit from the most enclosing cycle.

8. The choice action. The choice action is of the form:

```
[
   | labels :
        actions
   | labels :
        actions
        ...
   | * :
        actions
]
```

The choice action tries to match the current input token to one of the labels. The actions associated with the matched label are executed; if no label is matched, the final alternative, labelled by the star '*', is executed.

This completes a summary of the actions in SL. The actions in S/SL are the same with the addition of calls to semantic operations.

Now an example S/SL rule is given which uses a count stack. Suppose a stack of counters is defined in S/SL as follows:

```
mechanism count:
   CountPushSymbolDimension  % new counter gets value from
                                 symbol table
   CountPop                  % delete counter
   CountIncrement            % add 1 to top counter
   CountDecrement            % subtract 1 from top counter
   CountChoose >> number;    % check the number of parameter
```

This example rule handles a list of actual parameters for a procedure call in a language such as Pascal.

```
 1  ActualParameterList:
 2     CountPushSymbolDimension
 3     {
 4        @HandleActualParameter
 5        CountDecrement
 6        [
 7           | 'rightParenthesis' :
 8              >
 9           | 'comma' :
10        ]
11     }
12     [ CountChoose
13         | zero :
14         | * :
15             # WrongNumberActuals
16     ]
17     CountPop;
```

Suppose the Pascal procedure call is

```
P (x,y)
```

The procedure's name P is accepted by the S/SL program and becomes the current symbol of interest. Then the left parenthesis is accepted and finally the 'ActualParameterList' rule is called. Line 2 finds the declared number of parameters of P and pushes this number onto the count stack. Then the loop (lines 3 through 11) processes the list of actuals, decrementing the count for each actual. 'HandleActualParameter' will accept 'x' the first time through the loop and 'y' the second time.

Then lines 12 through 16 print an error message if the wrong number of actual parameters was supplied. Line 17 pops the counter that was pushed on line 2.

2.3 The reasons for using S/SL in this project

S/SL has been used in implementing three compilers: Speckle (a PL/1 subset)(10), Toronto Euclid(11), and PT (a Pascal subset)(12). It has also been used to implement an S/SL processor. S/SL has been found to be powerful, convenient and expressive, but it also is quite readable and maintanable.

The S/SL language synthesizes several programming concepts. The following related notations have been strongly influential:

(a) BNF and regular expressions

(b) Syntax charts and semantic charts

(c) Recursive descent compilers

(d) Separable transition diagrams

(e) Table-driven coding

(f) Data encapsulation technique.

This implies that S/SL captures such a large fraction of the power of these notations, while itself remaining very simple.

Besides all the significant characteristics of the S/SL language described above, the main reason we used the S/SL program to serve as a recursive descent parser for Simula prettyprinter (SPP) was that a parser for an Euclid prettyprinter (EPP) was written in S/SL. Thus, in implementing the SPP, the main concern was the meaning of the semantic mechanisms and the

modifications to the scanner. The meaning of the semantic mechanisms was important in constructing the parser portion in S/SL for Simula. There was no need for directly being concerned with implementation details which were written in Pascal. EPP's lexical scanner needed modification to make a new lexical scanner for Simula.

Chapter 3

The general description of the Simula prettyprinter

3.1 The functions of the Simula prettyprinter

Simula prettyprinter(SPP) is a program which serves as a full parsing source text formatter. SPP parses and formats the entire Simula language, token by token. Thus, it is able to format type definitions, parameter lists, expressions, variable references, etc. SPP allows the programmer to type source text without any regard to formatting, for both initial text entry and text modification. Its tasks include the following functions:

a. Insymbol - works as a lexical scanner which reads tokens from the input source text and classifies them.

b. Blankline - handles the placement of blank lines in the output.

c. PushConstPos - pushes integer constant 1 onto the position stack.

d. PushPos - accepts a parameter which is a position value and pushes it onto the position stack.

e. PushMark - It accepts a parameter which is a mark record and pushes it onto the mark stack.

f. PopPos - pops the top value from the position stack.

g. PopMark - pops the top value from the mark stack.

h. Mark - marks some tokens. It marks the tokens so that at some later points all tokens starting there will be compressed.

i. Skip - skips some unexpected tokens when syntax errors are encountered.

J. Setup - sets up a starting position for a token in the output.

k. Movecmnts - moves the positions of some comments if necessary.

l. Indent - indents some fixed amount of spaces for some tokens.

m. Ccompress - compresses some tokens if they can be arranged into one line.

n. Unload - unloads and prints a set of tokens which will not be compressed later or whose length is more than one line.

3.2 The data structure of the SPP

The data structure of the Simula prettyprinter is illustrated in Figure 2. It consists of a set of nodes, each of which is a line record which includes downlink, rightlink, start position, text, and other fields. Each node only contains a text of 12 characters. Thus if the length of a token is over 12 characters, as many nodes as necessary are cdreated, with their rightlink fields linking them together. The downlink field is used to connect between different tokens. Gfirstp is a pointer to the first token in a set of tokens which have not been released. Gcurp is a pointer to the current token. Gnextp is a pointer to the token which is next to the current token. Gcomp

is a pointer to the token from which a set of tokens needs to be compressed later. The mark is a special kind of node whose pointer field points to the first token of a set of tokens which will be compressed later.

Using the above data structure, the SPP compresses tokens in a very straightforward way. It first marks a possible starting token from which a set of tokens may be compressed later. Then the SPP sets up the starting position for each token that follows including a possible end token. It assumes that no syntactic structure will fit in one line. After that, SPP compresses this set of tokens into one line if they can fit; otherwise, they will be arranged into several lines as necessary. The example in Figure 3 shows a basic idea of how SPP compresses the tokens.

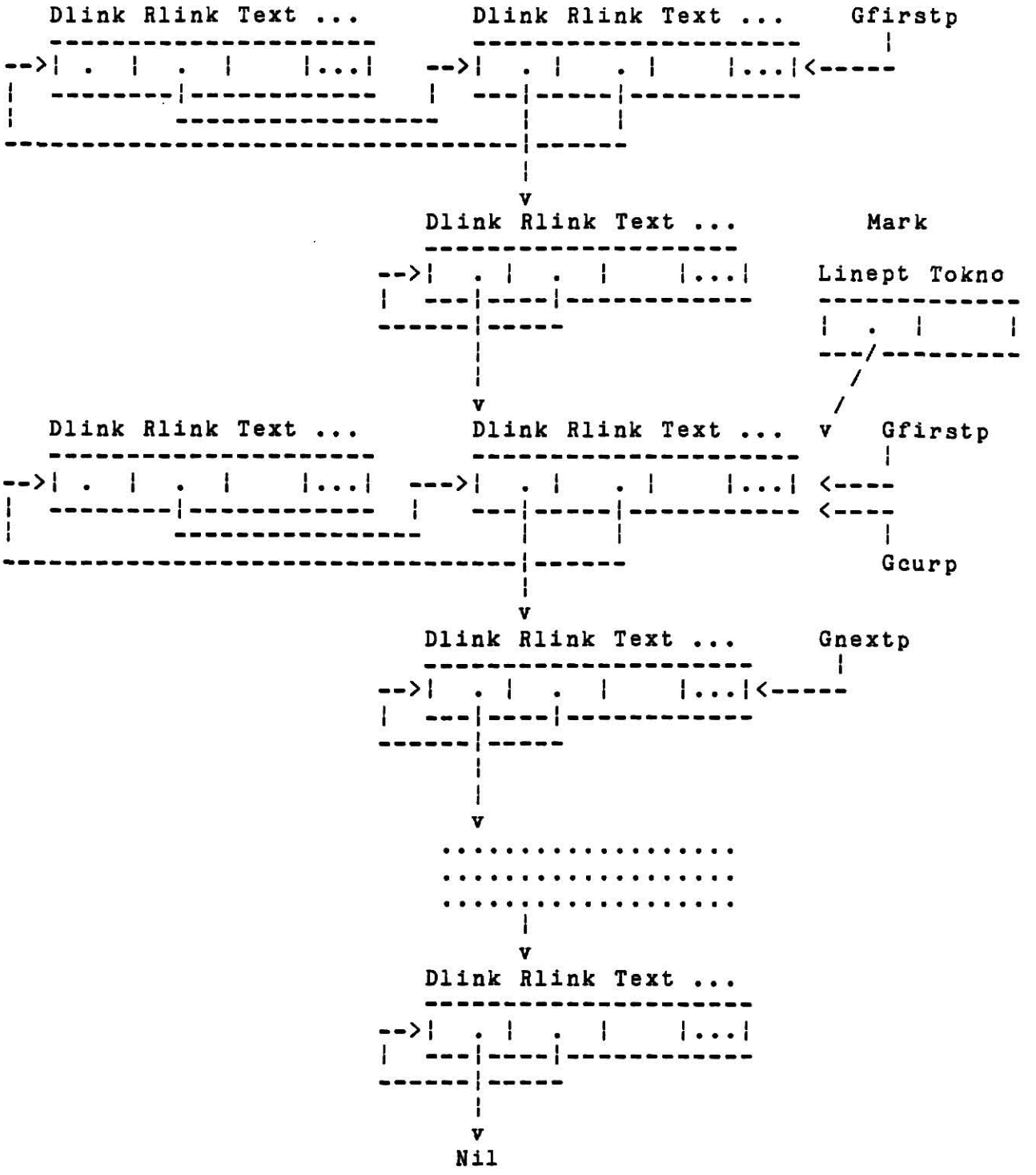

Figure 2. The data structure of SPP

before compression

after compression

(1).

```
Begin
  a := 1 ;
  b := 2 ;
  c := 3
End
```

```
Begin a := 1 ; b := 2 ; c := 3 End
```

(2).

```
Begin
  read (a) ;
  If X
  Then Y
  Else Z ;
  write (a)
End
```

```
Begin
  read (a)
; If X Then Y Else Z
; write (a)
End
```

Figure 3. Example of compressing tokens

3.3 The Indentation style used in Simula prettyprinter

The indentation style used was designed by Dr. Rodney M. Bates and is based on the observation that most of the syntatic structures are 'fences', with n 'post' and n-1 'panels' between them. The posts are always single symbols in Simula language such as Begin, End, If, Then, etc. The panels are fragments of Simula program which can be strings of arbitrarily length, such as declarations, statements, expressions, etc.

The fence model suggests that all the post symbols of a single structure that are related to one another should be vertically aligned. One of the fascinating things in the SPP is the placement of the character ";". The semicolon is treated as a post symbol of the Simula compound statement. Thus the Begin, End, and semicolon, etc. should be vertically aligned in a related structure. The example in Figure 4 shows how a compound statement is formated according to the fence model.

In this example, the leading semicolon shows clearly where each constituent statement of the compound statement should be positioned, even though the if statement occupied three lines, while the others occupy only one each. Thus it makes the format of a text more readable. Furthermore, not only can SPP format statements, but it can format basic primitives such as expressions. The formatting rule used on an expression is the same one used on a semicolon. The example in Figure 5 illustrates how SPP works neatly on semicolons and expressions. Note that the example is in uncompressed formatting, the whole expression and subexpressions are usually compressed later.

```
Begin
  A := X
; If A > 0
  Then A := A - 1
  Else A := A + 1
; B := A
End
```

Figure 4. Example of formatting compound structure by fence model

before formatted

```
(1).
     a := b+c*d;
     e := x/y-z;
```

after formatted

```
  a
  := b
     + c
       * d
; e
  := x
     / y
     - z
;
```

```
(2).
     x := 2*(a+b);
     y := (a+b*c)-3;
```

```
  x
  := 2
     * ( a
         + b
       )
; y
  := ( a
       + b
         * c
     )
     - 3
;
```

Figure 5. Example of formatting expressions

Chapter 4

Simula prettyprinter implementation

4.1 The process Overview

Implementing the Simula prettyprinter has been a fantastic and unique experience. The entire work was complicated but very interesting. First of all, the syntax of the Simula language needed to be described. This work took more time than was expected because there were no complete definitions for Simula's syntax. The author had to describe the Simula's syntax by modifying the syntax of Algol 60(9) which is the subset of Simula and combining the syntax of Algol 60, and that of the extension part of Simula(14). It was also necessary to know the exact meaning of the semantic routines implemented in EPP before constructing the S/SL program. Then, using the syntax of Simula as a framework for the parser, The semantic mechanisms were defined in the S/SL program to complete the parser portion for Simula prettyprinter. Furthermore, the lexical scanner in Euclid prettyprinter was modified for Simula programs due to the differences in reserved words and special characters between Euclid and Simula.

After the above programs had been finished, the next step was the implementation. First, the S/SL program was translated into a S/SL table(or a S-code) using the S/SL assembler. Then, the S/SL table in the Euclid prettyprinter was replaced with the newly developed S/SL table to make a new Simula

prettyprinter(SPP) program. The S/SL table could in turn be accessed by a procedure called a table walker or S-code interpreter in the SPP to execute the entire S/SL program. The final step was to compile and run the SPP written in Pascal on the Interdata 8/32 machine under Unix V7. The entire process of the project can be represented by a diagram in Figure 6. The following sections in the chapter will describe the process in detail.

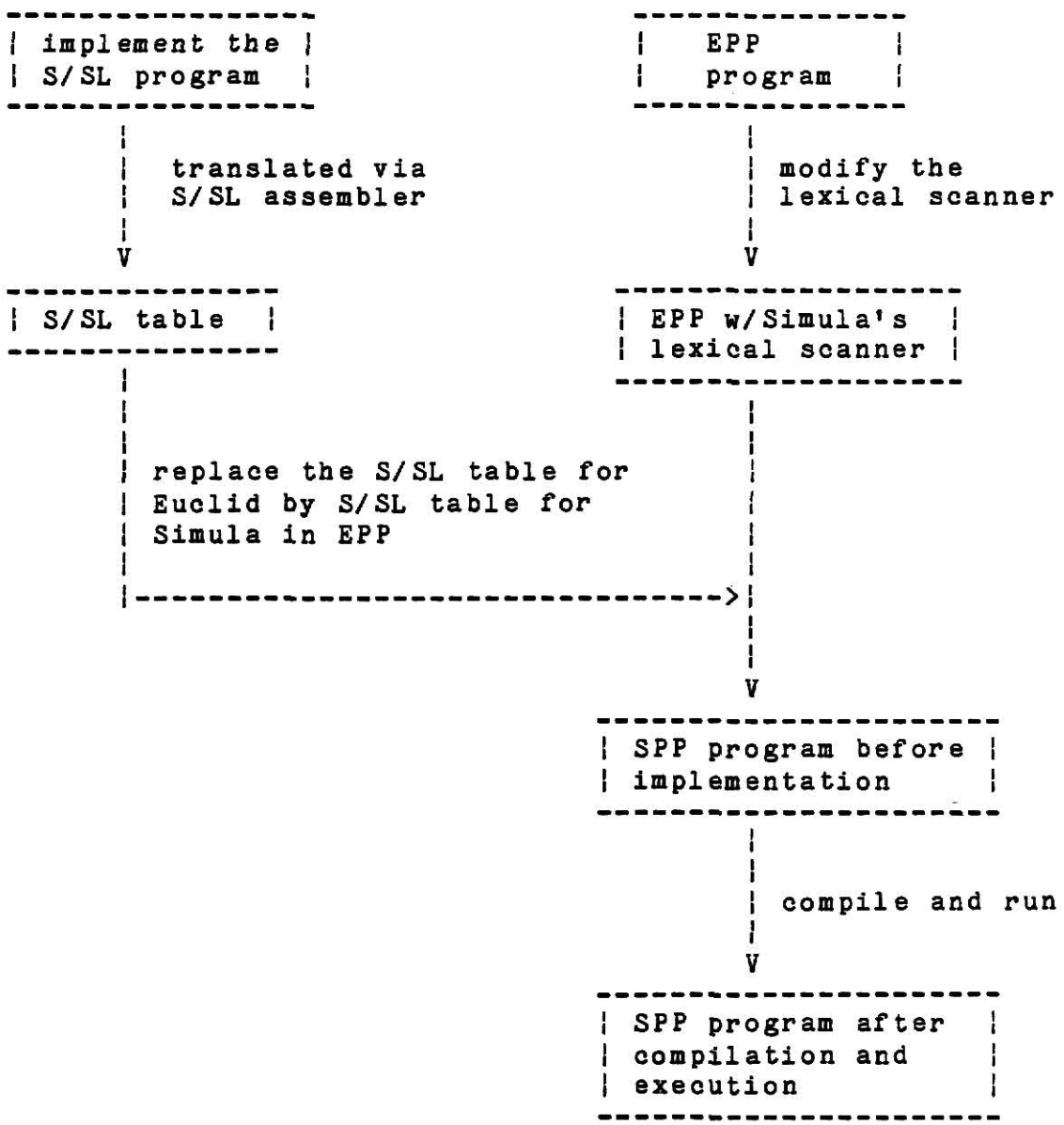

Figure 6. The process of generating an SPP

4.2 The Implementation of S/SL program

One of the main reasons for using the Syntax/Semantic Language(S/SL) program to serve as a recursive descent parser for SPP was that the parser for Euclid prettyprinter(EPP) was written in S/SL. Thus, with the existing semantic routines in the EPP, only the meaning of those routines needed to be understood to implement the S/SL program for SPP.

Because S/SL is a variable-free language, two important semantic mechanisms, namely, the stacks of positions and marks, exist to handle the problems of local variable and parameter passing in the S/SL program. In addition, the semantic routines such as the Ccompress, SetupPos, Indent, etc. are imperative to implement the S/SL program.

In the following paragraphs, an example is used to show how the stacks of positions and marks efficiently handle the problems of local variables and parameter passing, and how the other semantic routines are used in the S/SL program. To simplify problems, the example in Figure 7 uses only the syntactic structure of the If statement in Simula. It will be shown how a syntactic structure is formatted at first, and how it is formatted after being compressed.

In the example, an assumption is made that the S/SL program is already in a rule called statements. Then, an input choice is encountered. Prior to setting up the position for the token 'if', it is necessary to indent 'if' to position it right after a semicolon. A simple way to do this is to fetch the top value of the positions stack, which is the position value for the

semicolon passed from the caller rule, and use it as a parameter to an SPP routine called Indent. The indented position value returned from this routine is in turn pushed onto the top of the position stack. After that, we set up the position for 'if' by fetching the top element of the position stack and using it as a parameter to an SPP routine called Setup. These processes explain how the position stack, positions and the formatting routines (such as Indent and Setup) work together to set up positions for tokens. Namely, it shows how a syntactic structure is formatted at first before being compressed later.

Next, the rule called Ifstatement will be invoked to handle the body of the If statement. Some problems must be considered before this is done. First, the token 'if' will eventually have to be compressed with the body of the If statement after returning from a call to the rule Ifstatement, regardless of whether they can be fitted in one line. Secondly, an expression is usually placed right after the 'if' operator, if both can be fitted in one line. This implies that another compression will be executed within the routine Ifstatement. The solution to these problems is to push the current token mark, whose pointer field points to the token 'if', onto the mark stack twice, prior to setting up the position for 'if'. The first current token mark will be fetched and used as a parameter to a semantic routine called Ccompress. This checks whether the whole If statement(including 'if') can be fitted in one line. This is done after the second mark has been popped from the mark stack when the routine Ifstatement was returned. The second current

token mark pushed onto the mark stack is used as a parameter to the routine Ifstatement, and then it serves as a starting point from which some tokens might be compressed later. In Figure 7.(3), two different possible formats resulting from the above process are shown. Apparently, the process described above explains us how the mark stack and marks are used with the routine Ccompress to compress tokens. Namely, it deals with how a syntactic structure is formatted after being compressed.

In order to help solve problems such as local variables and parameter passing, it is necessary to push a position value onto the top of the position stack. This serves as an additional parameter besides the other, which is the top value of the mark stack, to a called rule. Thus, these two values not only serve as parameters to a called rule, but they serve as two variables local to that called rule. As the example showed in Figure 7, it pushes the position value for 'if' onto the position stack and the current token mark, whose pointer field points to 'if', onto the mark stack, and passes both of them as parameters to a rule called Ifstatement. Later, within this rule, these two values will be used to set up positions for tokens, compress tokens, and pass them as parameters to other called rules. The process described above explores how two semantic mechanisms, i.e., the mark stack and the position stack are used to pass both the position value and the mark value to a called rule. Then this rule uses both values as its local variables to format its substructures. One thing to be aware of is whenever a caller is returned from a rule, the top value of both the position stack and the mark stack is always popped, because these two values are

only meaningful to a called rule. Figure 7.(1) shows an original, unformatted program. Figure 7.(2) shows how this program is formated at first. Figure 7.(3) shows two possible formats after this program is compressed in accordance with the length of the structure.

The interfaces to these semantic mechanisms were defined in the S/SL program. But the implementation of these semantic mechanisms was done in the Pascal program. With the help of these semantic operations, the Simula parser, written in S/SL was able to parse and reformat the input Simula program in a straightforward manner.

Conventionally, a small letter 'o' was put as the first letter of each instruction name, for example, 'Ccompress' becomes oCcompress to avoid clashes with other names in the Pascal program. Some other examples are oSetupPos0, oPopMark, oPopPos, oSwapmarks, and so on.

In the beginning, it seemed very difficult, not only to construct a parser from the syntax descriptions without any ambiguities and mistakes, but also to understand how all these semantic mechanisms could be integrateed. But after a little practice in constructing the parsing rules and experience in using the semantic mechanisms, the entire process of defining the semantic mechanisms with the parser fell into certain 'patterns', and it was then very easy to go on to each rule.

(1). The original, unformatted If statement

```
                    ;
If X
  Then Y
  Else Z      ;
```

(2). The formatted If statement at first before compression

```
; If X
  Then Y
  Else Z
;
```

(3). Three possible formats of If statement after compression

(a).

```
; If X Then Y Else Z
;
```

(b).

```
; If X
  Then Y
  Else Z
;
```

Figure 7. Example of using two semantic mechanisms: the position stack and the mark stack

4.3 The modification of EPP's lexical scanner

The next process in implementing the SPP was to modify the existing lexical scanner in EPP into lexical scanner for Simula. The function of the lexical scanner is to read the source program one character at a time and to translate it into a sequence of primitive units called tokens. Keywords, identifiers, constants, and operators are examples of tokens.

Due to different keywords defined in Euclid and Simula, it was necessary to replace the keywords used in Euclid with those used in Simula. Also there are some special characters in Simula which Euclid does not have and vice versa. For example, ':-'(denotes), '**'(exponent), and '=='(reference equal), etc., which are tokens in Simula but not in Euclid, had to be added. Operators, like '=>'(label or imply), and '..'(double periods), etc., used in Euclid were deleted. Further, Simula uses the keyword 'comment' at the beginning of comments and the symbol ';' as comment ending delimiter. Thus, 'comment' and ';' had to be added while the symbols '{' and '}' which delimit comments in Euclid needed to be removed.

4.4 The translation and the inclusion of the S/SL program

The S/SL program needed to be included into the SPP to be accessed and executed after it was implemented. Since the S/SL program was not Pascal-coded, it was necessary to translate it into S-code form (or S/SL table) via a program called the S/SL assembler. This transformed S/SL program contains Pascal

declarations for the S/SL table and the constants defining the table operation codes, input tokens, output tokens, error codes, and semantic operation codes used in the table.

The translated S/SL program refers to an S/SL table because the sequence of numbers in this table are stored in a Pascal array. This table is included in the declaration part of the SPP. Later, a procedure in SPP calls a table walker or S-code interpreter, which accesses this table and executes its instructions to carry out the actions of the S/SL program.

4.5 The implementation and running of the SPP

The S/SL assembler and the S-code interpreter used in this project automatically produce Hartmann(15) style recovery from syntax errors. Thus, the error recovery problems in the Simula parser were avoided. This allows the Simula prettyprinter to format a large variety of incorrect constructs and enables it to continue formatting correct text beyond those constructs. Actually, by combining steps 2, 3 and 4, we obtained a Simula prettyprinter program written in Pascal. This program was then compiled and run as usual, the result being a Simula prettyprinter.

Running the SPP is simply done by typing the word SPP followed by an input file name which is intended to be prettyprinted. The output will then be stored under the the same file name, but with an extention SPP. Each time users run the SPP, a report is displayed giving the number of lines in the

input file, number of lines in the output file, the number of Simula tokens in the fsle(excluding comment tokens), the number of tokens skipped, and how many heap records were created. A typical report looks like this:

```
 93  input lines
169  output lines
718  tokens
  0  skipped
140  heap records
```

The number of tokens skipped shows an estimate of the difficulty SPP had with formatting. Small numbers, i.e. 0 .. 3, mean the formatting is probably no problem. Larger numbers mean the SPP found some syntax errors and skipped them. Since the semantic routines use the Hartmann error recovery, SPP will not reformat sections of the program in which an error is detected. This implies that the user can run the SPP on a newly created or modified Simula program even before compiling it.

Chapter 5

Conclusion

In general, the experience of writing the Simula prettyprinter using S/SL program as the parser has been an excellent one. The main feature of the Simula prettyprinter is that it uses an S/SL program to serve as a recursive descent parser portion of the SPP. This implies that we can define the meaning of semantic mechanisms without directly being concerned with implementation details.

The S/SL program included in SPP (appendix 1) consists of 1870 lines. SPP itself (appendix 2) contains 2432 lines. The author has tested over fifty sample Simula programs as inputs to run the SPP successfully. One of the sample input program is shown in appendix 3 contains 63 lines. The output program generated from it contains 125 lines, and is shown in appendix 4. The SPP took 40.5k bytes of memory and 8.0 seconds to run this sample program.

Due to the nature of the data structure of SPP, the amount of space needed to run the program does not increase with the length of the source program but rather with the depth of nesting of syntactic structures in the source program. The table produced by S/SL together with the table walker are observed to be quite small, better than or comparable to other techniques such as writing in a high level language or using LR(k).

References

1. Hueras, Jon and Ledgard, Henry; An Automatic Formatting Program for Pascal; SIGPAN Notices, Vol.12, No.7, July 1977, pp. 82-84.
2. Clifton, Mitchell H.; A Technique for Making Structured Programs More Readable; SIGPLAN Notices, Vol.13, No.4, April 1978, pp. 58-63.
3. Ramsdell, John; Prettyprinting Structured Programs with Connector Lines; SIGPLAN Notices, Vol.14, No.9, Sept. 1979, pp. 74-75.
4. Hearn, Anthony C. and Norman, Arthur C.; A One-Pass Prettyprinter; SIGPLAN Notices Vol.14, No.12, Dec. 1979, pp. 50-58.
5. Bates, Rodney M.; A Pascal Pretty-Printer with a Different Purpose; SIGPLAN Notices, vol. 16, No.3, March 1981, pp. 10-17.
6. Unix Time-Sharing System: Unix Programmer's Manual; Seven Edition, Volume 2A; Bell Telephone Laboratories, Inc., Murray Hill, New Jersey, Jan. 1979.
7. Lin, Wun-Jen; Euclid Pretty-Printer Using Pascal; The master report, 1983, Kansas State University, Manhattan, Kansas.
8. Cordy, J. R., Holt, R. C. and Wortman, D. B.; An Introduction to S/SL: Syntax/Semantic Language; ACM Transactions on Programming Language and Systems, Vol.4, No.2, April 1982, pp. 149-178.
9. Ghezzi, Carlo and Jazayeri, Mehdi; Programming Language

Concepts; Synapse Computer Corporation, 1982.

10. Miceli, J.; Some Experiences with a One-man Language; Computer Systems Research Group, University of Toronto, Technical Note 13, Dec. 1977.
11. Holt, R. C., Wortman, D. B., Cordy, J. R., and Growe, D. R.; The Euclid Language: A Progress Report; Proceedings of the ACM Natinal Conference, Dec. 1978.
12. Rosselet, J. A.; A Pascal Subset; The Master Thesis, University of Toronto, Jan. 1980.
13. Dahl, Ole-Johan, Myhrhaug, Bjorn and Nygaard, Kristen; SIMULA 67 Common Base Language; Norwegian Computing Center, 4ed., 1982
14. Bauer, F. L., Green, J., Katz, C., Kogon, R., Naur, P., Samelson, K., Wegstein, J. H., Wijngaarden, A. Ven, and Woodger, M.; Revised Report on the Algorithmic Language ALGOL 60; Numerische Mathematik 4, 1963. pp.420-453, Communications ACM, Jan. 1963. pp. 1-17, and Computer Journal Vol.5, No.4, Jan. 1963. pp. 349-367.
15. Hartmann, A. C.; A Concurrent Pascal Compiler for Minicomputers; Springer Verlag, 1977.

Appendix 1

Simula prettyprinter code

```
      1
      2 (*************************************************
c     3  *                                               *
c     4  *      SIMULA prettyprinter     VERSION 1.0     *
c     5  *      Jung-Juin Chen                           *
c     6  *       November 1985                           *
c     7  *                                               *
c     8  *************************************************)
      9 (% PASCAL32 := TRUE )
     10 (% PASCAL := NOT PASCAL32)
     11 (% ASCII := TRUE)
     12 (% EBCDIC := NOT  ASCII)
     13 (% VAX := FALSE)
     14 (% PREP := TRUE)
     15 (% SINGLEPRECISION := TRUE)
     16 (% MULTPRECISION := NOT SINGLEPRECISION)
     17 (%IF PASCAL32)
     18 (##########
c    19 #  PREFIX  #
c    20 ##########)
     21   type integer32 = integer                                          *     S
     22
     23 ; const maxint32 = maxint                                          *     S
     24
     25 (% IF MULTPRECISION)
     26
     27 ; type integer = shortinteger                                      S     S
     28
     29 ; const maxint = maxshortint                                       S     S
     30
     31 (% END MULTPRECISION)
     32
     33 ; const nl = '(:10:)'                                             *
     34   ; ff = '(:12:)'                                                 *
     35   ; cr = '(:13:)'                                                 *
     36   ; em = '(:25:)'                                                 *
     37
     38 ; const pagelength = 512                                          *
     39
     40 ; type page = array [ 1 .. pagelength ] of char                    S    38.    S
     41
     42 ; const linelength = 132                                          *
     43
     44 ; type line = array [ 1 .. linelength ] of char                    *    42     S
     45
     46 ; const idlength = 12                                             *
     47
     48 ; type identifier = array [ 1 .. idlength ] of char               *    46     S
     49
     50 ; type pfxfile = 1 .. 5                                           *
     51
     52 ; type filekind = ( empty , scratch , ascii , seqcode , concode )  *     S     *     *     *     *
     53
     54 ; type fileattr                                                    *
```

```
    56          kind : filekind                                          *    52
    57        ; addr : integer                                           *    27
    58        ; protected : boolean                                      *     S
    59        ; notused : array [ 1 .. 5 ] of integer                    *    27
    60        end
    61
    62 ; type iodevice                                                   *
    63     = ( typedevice , diskdevice , tapedevice , printdevice       *     *     *     *
    64       , carddevice )                                             *
    65
    66 ; type iooperation = ( input , output , move , control )          *     S     S     *     *
    67
    68 ; type ioarg = ( writeeof , rewind , upspace , backspace )        *     *     *     *     *
    69
    70 ; type ioresult                                                   *
    71     = ( complete , intervention , transmission , failure , endfile *     *     *     *     *
    72       , endmedium , startmedium )                                *     *
    73
    74 ; type ioparam                                                    *
    75     = record
    76          operation : iooperation                                  *    66
    77        ; status : ioresult                                        S    70
    78        ; arg : ioarg                                              *    68
    79        end
    80
    81 ; type taskkind = ( inputtask , jobtask , outputtask )           *     *     *     *
    82
    83 ; type argtag = ( niltype , booltype , inttype , idtype , ptrtype ) *     *     *     *     *     *
    84
    85 ; type pointer = ^ boolean                                        *     S
    86
    87 ; type passptr = ^ passlink                                       *     *
    88
    89 ; type passlink                                                   *
    90     = record
    91          options : set of char                                    *     S
    92        ; filler1 : array [ 1 .. 7 ] of integer                    *    27
    93        ; filler2 : boolean                                        *     S
    94        ; resetPoint : integer                                     *    27
    95        ; filler3 : array [ 1 .. 11 ] of pointer                   *    85
    96        end
    97
    98 ; type argtype                                                   *
    99     = record
   100          case tag : argtag                                        *    83
   101          of niltype , booltype : ( bool : boolean )              83    83     *     S
   102          ; inttype : ( int : integer )                           83     *    27
   103          ; idtype : ( id : identifier )                          83     *    48
   104          ; ptrtype : ( ptr : passptr )                           83     *    87
   105        end
   106
   107 ; const maxarg = 10                                              *
   108
   109 ; type arglist = array [ 1 .. maxarg ] of argtype                *   107    98
   110
   111 ; type argseq = ( inp , out )                                     *     *     *
```

```
112
113 ; type progresult                                                    *
114     = ( terminated , overflow , pointererror , rangeerror           *     *     *     *
115       , varianterror , heaplimit , stacklimit , codelimit , timelimit *     *     *     *     *
116       , callerror )                                                 *
117
118 ; procedure pfxread ( var c : char )                                 *     *     S
119
120 ; procedure pfxwrite ( c : char )                                    *   118     S
121
122 ; procedure open                                                     *
123     ( f : pfxfile ; id : identifier ; var found : boolean )         *    50   103    48     *     S
124
125 ; procedure close ( f : pfxfile )                                    *   123    50
126
127 ; procedure pfxget                                                   *
128     ( f : pfxfile ; p : integer ; var block : univ page )           125    50     *    27     *    40
129
130 ; procedure put ( f : pfxfile ; p : integer ; var block : univ page )   S   128    50   128    27   128    40
131
132 ; function length ( f : pfxfile ) : integer                          *   130    50    27
133
134 ; procedure mark ( var top : integer )                               S     *    27
135
136 ; procedure release ( top : integer )                                S   134    27
137
138 ; procedure identify ( header : line )                               *     *    44
139
140 ; procedure accept ( var c : char )                                  *   120     S
141
142 ; procedure display ( c : char )                                     *   140     S
143
144 ; procedure notused                                                 59
145
146 ; procedure notused2                                                *
147
148 ; procedure notused3                                                *
149
150 ; procedure notused4                                                *
151
152 ; procedure notused5                                                *
153
154 ; procedure notused6                                                *
155
156 ; procedure notused7                                                *
157
158 ; procedure notused8                                                *
159
160 ; procedure notused9                                                *
161
162 ; procedure notused10                                               *
163
164 ; procedure run                                                     S
165     ( id : identifier                                             123    48
165     ; var param : arglist                                           *   109
                                                                       44    27
```

```
    168     ; var result : progresult                                          *   113
    169     )
    170
    171 ; program spp ( param : line )                                          *   166   167
    172 {% END PASCAL32}
    173 {% IF PASCAL}
    174     program spp ( input , output , initfile , summary )                171    66    66      *     *
    175
    176 {% END}
    177
    178     ; include 'spp.scd'                                                *
    179       include 'spp.dcl'                                                178
    180 { Declarations used by semantic routines and Interpreter }
    181       {Primitive table Operations:
c   182
c   183
c   184         These will remain the same independent of the
c   185         S/SL program and its semantic actions. }
    186       const firsttableOperation = 0                                    *
    187
    188     ; const firstPrimitiveOperation = 0                                *
    189       ; oCall = 0                                                      *
    190       ; oReturn = 1                                                    *
    191       ; oRepeat = 2                                                    *
    192       ; oMerge = 3                                                     *
    193       ; oInput = 4                                                     *
    194       ; oInputChoice = 5                                               *
    195       ; oEmit = 6                                                      *
    196       ; oError = 7                                                     *
    197       ; oChoiceEnd = 8                                                 *
    198       ; oRecovCall = 15                                                *
    199       ; oRecovInput = 16                                               *
    200       ; oRecovInputChoice = 17                                         *
    201       ; oRecovChoiceFailure = 13                                       *
    202       ; lastPrimitiveOperation = 18                                    *
    203         {Semantic Operations:
c   204
c   205          These will be different for each pass.  The
c   206          semantic operations are implemented in the
c   207          semantic module. }
    208       ; firstSemanticOperation = 19                                    *
    209       ; firstSimpleOperation = 19                                      *
    210       ; lastSimpleOperation = 39                                       *
    211       ; firstParameterizedOperation = 40                               *
    212       ; lastParameterizedOperation = 45                                *
    213       ; firstChoiceOperation = 45                                      *
    214       ; lastChoiceOperation = 45                                       *
    215       ; firstEmittingOperation = 45                                    *
    216       ; lastEmittingOperation = 45                                     *
    217       ; lastSemanticOperation = lastEmittingOperation                  *   216
    218       ; lasttableOperation = lastSemanticOperation                     *   217
    219       ; dummyLine = 0                                                  *
    220       ; setSize = 7                                                    *
    221
    222     ; type SetArray = array [ 0 .. setSize ] of shortinteger           *   220     5
                                                                               *
```

```
224       ; setofsymbol = set of symboltyp                                  *   223
225
226     ; type CallStackEntry                                               *
227         = record
228             sett : setofsymbol                                          *   224
229                                 {Dynamic set of syntax recovery tokens}
230           ; returnAddress : 0 .. tableSize                              *     *
231           end {allStackEntry}
232
233     ; const callStackSize = 127                                         *
234       ; maxLineNumber = 29999                                           *
235         {Error Output Interface }
236       ; firstErrorCode = 0                                              *
237       ; eNoError = 0                                                    *
238       ; eEndOfErrors = 1                                                *
239       ; firstTrueErrorCode = 2                                          *
240       ; firstParserErrorCode = 21                                       *
241       ; eSyntaxError = 21                                               *
242       ; firstFatalErrorCode = 22                                        *
243       ; ePrematureEndOfFile = 22                                        *
244       ; eCallStackOverflow = 23                                         *
245       ; eExtraneousProgramText = 24                                     *
246       ; ePosStackOverflow = 25                                          *
247       ; ePosStackUnderflow = 26                                         *
248       ; eMarkStackOverflow = 27                                         *
249       ; eMarkStackUnderflow = 28                                        *
250       ; eDuplicateSetup = 29                                            *
251       ; eNotSetup = 30                                                  *
252       ; eSetupUnmatchedToken = 31                                       *
253       ; lastErrorCode = 31                                              *
254
255     ; type ErrorCodes = firstErrorCode .. lastErrorCode                 *   236   253
256
257     ; const maxErrors = 20                                              *
258
259         {Input Interface }
260
261     ; const firstInputToken = - 1                                       *
262
263     ; const tSyntaxError = - 1                                          *
264
265     ; const lastInputToken = tXor                                       *     *
266
267     ; const leftmargin = 1                                              *
268       ; rightmargin = 72                                                *
269       ; MaxPosStack = 127                                               *
270       ; MaxMarkStack = 200                                              *
271       ; shiftpos = 36                                                   *
272       ; indentamt = 2                                                   *
273       ; fraglength = 12                                                 *
274       ; rwmax = 80                                                      *
275       {% IF ASCII}
276       ; ht = 9 {HORIZONTAL TAB}                                         *
277       ; ff = 12 {FORM FEED}                                            34
278       {% END ASCII}
279       {% IF EBCDIC}
```

```
280       ; ht = 5 {HORIZONTAL TAB}                                          276
281       ; ff = 12 {FORM FEED}                                              277
282       {% END EBCDIC}
283       ; multmct                                                           *
284         = 2 {NUMBER OF INTEGERS IN A MULTIPLE PRECISION INTEGER}
285
286     ; type positiontyp = integer                                          *    27
287       {% IF SINGLEPRECISION}
288       ; multint = integer                                                 *    27
289       {% END SINGLEPRECISION}
290       {% IF MULTPRECISION}
291       ; multint = array [ 1 .. multmct ] of integer                      288   283    27
292         {MULTMCT IS MOST SIGNIFICANT}
293       {% END MULTPRECISION}
294       ; setofchar = set of char                                           *     S
295       ; linerecptr = ^ linerec                                            *     *
296       ; linerec                                                           *
297         = record
298             dlink , rlink : linerecptr                                    *     *   295
299           ; tokno , aggsize : multint                                     *     *   291
300           ; symbol : symboltyp                                            *   223
301           ; startpos : positiontyp                                        *   286
302           ; length : positiontyp                                        132   286
303           ; newline : boolean                                             *     S
304           ; skip : boolean                                                *     S
305           ; sepafter : boolean                                            *     S
306           ; moveable : boolean                                            *     S
307           ; text                                                          S
308             : {% IF PASCAL}
309               packed {% END } array [ 1 .. fraglength ] of char           *   273     S
310         end
311       ; markrec = record linep : linerecptr ; tokno : multint end         *     *   295   299   291
312       ; parproc                                                           *
313         = ( parconstant , parstc , parstcval , parstcelem                 *     *     *     *
314           , parvariable , parsetelem , parfactelem , parfactor            *     *     *     *
315           , parterm , parsmpexpr , parexpr , paractparam                  *     *     *     *
316           , parcompoundstmt , parxstmt , parlcase , parstmt , parfparm    *     *     *     *     *
317           , parsmptype , parproghead , parpfhead , parfldlist             *     *     *     *
318           , partype , parsyscomp , partypenamer , parblock                *     *     *     *
319           , parprogram )                                                  *
320       ; string12                                                          *
321         =  {% IF PASCAL }
322           packed {% END } array [ 1 .. 12 ] of char                       *     S
323       ; stringType = array [ 1 .. 100 ] of char                           *     S
324       {% IF PASCAL32}
325       ; logicalunitno = 0 .. 7                                            *
326       ; filestatus = ( closed , opforin , opforout )                      *     *     *     *
327       ; pageptr = ^ page                                                  *    40
328       ; text                                                            307
329         = record
330             fno : logicalunitno                                           *   325
331           ; pageno , bufp : integer                                       *     *    27
332           ; status : filestatus                                          77   326
333           ; eolnf , eoff : boolean                                        *     *     S
334           ; bufferp : pageptr                                             *   327
335           ; currentchar : char                                            *     S
```

```
   336           end
   337
   338     ;
   339       {% END PASCAL32}
   340   {The Syntax/Semantic table }
   341         {S/SL interpreter State }
   342       var processing : Boolean                                          *     S
   343       ; tracing : Boolean                                               *     S
   344       ; semanticizing : Boolean                                         *     S
   345       ; displaySyntaxErrors : Boolean                                   *     S
   346       ; tablePointer : 0 .. tableSize                                   *     *
   347       ; operation : firsttableOperation .. lasttableOperation          76   186   218
   348         {Line Counter }
   349       ; sourceLineNo : 0 .. maxLineNumber                               *   234
   350       ; keys : setofsymbol                                              *   224
   351       ; Lmark : markrec                                                 *   311
   352       ; choicetagMatched : boolean                                      *     S
   353         {The Call Stack:
c  354
c  355          The Call Stack implements Syntax/Semantic
c  356          Language procedure call and return.  In addition,
c  357          the Call Stack contains dynamic error recovery sets.
c  358          Each time an oCall operation is executed,
c  359          the table return address and the reachable set
c  360          for the oCall operator are pushed onto the
c  361          Call Stack.  When an oReturn is executed, the
c  362          return address and the reachable set are popped from the
c  363          Stack. An oReturn executed when the Call stack is
c  364          empty terminates interpretation.}
   365       ; callStackTop : 0 .. callStackSize                               *   233
   366       ; callStack : array [ 0 .. callStackSize ] of CallStackEntry      *   233   226
   367         {Parameterized And Choice Semantic Operation Values:
c  368
c  369          These are used to hold the decoded parameter value to
c  370          a parameterized semantic operation and the result
c  371          value returned by a choice semantic operation
c  372          respectively.                                  }
   373       ; parameterValue : integer                                        *    27
   374       ; resultValue : integer                                           *    27
   375       ; gError : 0 .. maxErrors                                         *   257
   376       { NEWLENGTH THRU NEWP USED BY INSYMBOL AND SUBROUTINES }
   377       ; newlength : integer                                             *    27
   378       ; pos , newstartpos , firstdotpos : positiontyp                   *     *     *   286
   379       ; innewline : boolean                                             *     S
   380       ; rbracksw , elipsissw : boolean                                  *     *     S
   381       ; cmntstate : ( nocomment , comment )                             *     *     *
   382       ; newsymbol : symboltyp                                           *   223
   383       ; newp : linerecptr                                               *   295
   384       ; i : integer                                                     *    27
   385       ; gfirstp , gtrailp , gcompp , gcomptrailp , gcurp , gnextp       *     *     *     *     *     *
   386         , gfreep                                                        *
   387         : linerecptr                                                  295
   388       ; setupState : ( unmatched , matchFailed , matched )              *     *     *     *
   389       ; gcursy : symboltyp                                              *   223
   390       ; gcurpos , gcurposnc , gpos , outpos : positiontyp               *     *     *     *   286
   391       ; gmark : markrec                                                 *   311
```

```
392       ; gtokno , outtokno , nltokno : multint                          *     *     *   291
393       ; gaggsize : multint                                             *   291
394       ; PosStack : array [ 1 .. MaxPosStack ] of positiontyp           *   269   286
395       ; MarkStack : array [ 1 .. MaxMarkStack ] of markrec             *   270   311
396       ; PosStackTop : 0 .. MaxPosStack                                 *   269
397       ; MarkStackTop : 0 .. MaxMarkStack                               *   270
398       {% IF MULTPRECISION}
399       ; linespace , multrightmargin : multint                          *     *   291
400       {% END MULTPRECISION}
401       ; rwptr : array [ 1 .. 13 ] of 1 .. rwmax                        *   274
402       ; rw                                                             *
403         : array [ 1 .. rwmax ]                                       274
404           of record
405                string : string12                                     *   320
406              ; symbol : symboltyp                                   300   223
407              end
408       ; rwct : integer                                                 *    27
409       ; letters , octaldigits , digits , hexdigits , lexchars          *     *     *     *     *
410         , throwaways                                                   *
411         : setofchar                                                  294
412       ; lineoutct , linerecct , tokenct , skipct : integer             *     *     *     *    27
413       {% IF PASCAL}
414       ; initfile , summary : text                                    174   174   328
415       {% END PASCAL}
416       {% IF PASCAL32}
417       ; input , output , initfile , summary : text                   174   174   414   414   328
418
419     ; procedure breakpnt                                               *
420       ; extern
421
422     ; procedure get ( var f : text )                                   *   132   328
423
424       ; begin
425           with f                                                     422
426           do if not eoff                                             333
427             then if fno in [ 0 , 1 , 3 , 4 , 5 , 6 , 7 ]             330
428               then begin
429                 case fno                                             330
430                 of 0 : accept ( currentchar )                        140   335
431                 ; 1 : pfxread ( currentchar )                        118   335
432                 ; 3 , 4 , 5 , 6
433                   : if status = opforin                              332   326
434                     then begin
435                       if bufp > pagelength                           331    38
436                       then begin
437                         pfxget ( fno - 2 , pageno , bufferp ^ )      127   330   331   334
438                       ; pageno := succ ( pageno )                    331     S   331
439                       ; bufp := 1                                    331
440                       end
441                     ; currentchar := bufferp ^ [ bufp ]              335   334   331
442                     ; bufp := succ ( bufp )                          331     S   331
443                     end
444                     else currentchar := ' '                          335
445                 end { CASE }
446               ; if currentchar = em                                  335    36
447                 then begin eoff := true ; currentchar := ' ' end     333     S   335
```

```
448                 else if currentchar = nl                                  335    33
449                   then begin eolnf := true ; currentchar := ' ' end       333     S    335
450                   else eolnf := false                                     333     S
451               end
452               else
453             else currentchar := ' '                                       335
454         end { get }
455
456       ; procedure open                                                    122
457           ( var f : text ; lu : logicalunitno ; mode : filestatus )       422   328     *   325     *   326
458
459         ; begin
460             with f                                                        457
461             do begin
462               eoff := false                                               333     S
463             ; eolnf := false                                              333     S
464             ; fno := lu                                                   330   457
465             ; if lu = 1                                                   457
466               then get ( f )                                              422   457
467             ; if lu in [ 3 , 4 , 5 , 6 , 7 ]                              457
468               then begin
469                 pageno := 1                                               331
470               ; status := mode                                            332   457
471               ; new ( bufferp )                                             S   334
472               ; if mode = opforin                                         457   326
473                 then begin bufp := pagelength + 1 ; get ( f ) end         331    38   422   457
474                 else if mode = opforout then bufp := 1                    457   326   331
475               end
476             end
477         end { OPEN }
478
479       ; function inputwindow ( var f : text ) : char                        *   457   328     S
480
481         ; begin
482             if f . currentchar in throwaways                              479   335   410
483             then inputwindow := ' '                                       479
484             else inputwindow := f . currentchar                           479   479   335
485         end { INPUTWINDOW }
486
487       ; function eof ( var f : text ) : boolean                             S   479   328     S
488
489         ; begin eof := f . eoff end                                       437   487   333
490
491       ; function eoln ( var f : text ) : boolean                            S   487   328     S
492
493         ; begin eoln := f . eolnf end                                     491   491   333
494
495       ; procedure write ( var f : text ; c : char )                         S   491   328   142     S
496
497         ; begin
498             with f                                                        495
499             do case fno                                                   330
500               of 0 : display ( c )                                        142   495
501               ; 2 : pfxwrite ( c )                                        120   495
502               ; 3 , 4 , 5 , 6 , 7
503                 : if status = opforout                                    332   326
```

```
504                 then begin
505                   if bufp > pagelength                                  331    38
506                   then begin
507                     put ( fno - 2 , pageno , bufferp ^ )                130   330   331   334
508                   ; pageno := succ ( pageno )                           331     5   331
509                   ; bufp := 1                                           331
510                   end
511                 ; bufferp ^ [ bufp ] := c                               334   331   495
512                 ; bufp := succ ( bufp )                                 331     5   331
513                 end
514             end
515         end ( WRITE )
516
517     ; procedure writeln ( var f : text )                                  5   495   328
518
519       ; begin write ( f , nl ) end                                      495   517    33
520
521     ; procedure close ( var f : text )                                  125   517   328
522
523       ; begin
524           with f                                                        521
525           do if fno = 2                                                 330
526             then pfxwrite ( em )                                        120    36
527             else if ( fno in [ 3 , 4 , 5 , 6 ] )                        330
528                     and ( status = opforout )                           332   326
529               then begin
530                 write ( f , em )                                        495   521    36
531               ; put ( fno - 2 , pageno , bufferp ^ )                    130   330   331   334
532               ; status := closed                                        332   326
533               end
534         end ( CLOSE )
535
536       {% END PASCAL32}
537     {% IF PASCAL }
538
539     ; function inputwindow ( var f : text ) : char                      479   521   328     5
540
541       ; begin
542         ; if f ^ in throwaways                                          539   410
543           then inputwindow := ' '                                       539
544           else inputwindow := f ^                                       539   539
545         end
546
547 {% END PASCAL }
548 {% IF MULTPRECISION}
549 { PROCEDURES FOR MULTIPLE PRECISION ARITHMETIC}
550
551     ; procedure multzero ( var result : multint )                       *   168   291
552
553       ; var i : 1 .. multmct                                            384   283
554
555       ; begin for i := 1 to multmct do result [ i ] := 0 end            553   283   551   553
556
557     ; procedure multplint                                               *
558         ( mop : multint ; iop : integer ; var result : multint )        *   291     *    27   551   291
559
```

```
550       ; var i , carryin , work : integer                                553     *     *    27
561         ; tempresult : multint                                           *    291
562
563       ; begin
564           carryin := iop                                                560   558
565         ; for i := 1 to multmct                                         560   283
566           do begin
567             if mop [ i ] > 0                                            558   560
568             then if ( mop [ i ] - maxint ) > ( - carryin )              558   560    29   560
569               then begin {PARANIOD CODING}
570                 work := mop [ i ] - maxint                              560   558   560    29
571               ; work := work - 2                                        560   560
572               ; work := work + carryin                                  560   560   560
573               ; tempresult [ i ] := work - maxint                       561   560   560    29
574               end
575               else tempresult [ i ] := mop [ i ] + carryin              561   560   558   560   560
576             else tempresult [ i ] := mop [ i ] + carryin                561   560   558   560   560
577           ; if mop [ i ] < 0                                            558   560
578             then if mop [ i ] + carryin >= 0                            558   560   560
579               then carryin := 1                                         560
580               else carryin := 0                                         560
581             else carryin := 0                                           560
582           end
583         ; result := tempresult                                          558   561
584         end { MULTPLINT } ;
585
586     ; procedure multsubtr                                                *
587         ( op1 , op2 : multint ; var result : multint )                   *     *   291   558   291
588
589       ; var i , work : integer                                          560   560    27
590         ; borrowin , borrowout : boolean                                 *     *     S
591         ; tempresult : multint                                          561   291
592
593       ; begin
594           borrowin := false                                             590     S
595         ; for i := 1 to multmct                                         589   283
596           do begin
597             if ( op1 [ i ] >= 0 ) and ( op2 [ i ] < 0 )                 587   589   587   589
598             then begin
599               borrowout := true                                         590     S
600             ; if op1 [ i ] - maxint > op2 [ i ]                         587   589    29   587   589
601               then begin {PARANCID CODING}
602                 work := op1 [ i ] - maxint                              589   587   589    29
603               ; work := work - 1                                        589   589
604               ; work := work - op2 [ i ]                                589   589   587   589
605               ; work := work - maxint                                   589   589    29
606               ; tempresult [ i ] := work - 1                            591   589   589
607               end
608               else tempresult [ i ] := op1 [ i ] - op2 [ i ]            591   589   587   589   587   589
609             end
610             else if ( op1 [ i ] < 0 ) and ( op2 [ i ] >= 0 )            587   589   587   589
611               then begin
612                 borrowout := false                                      590     S
613               ; if op1 [ i ] + maxint < op2 [ i ] - 1                   587   589    29   587   589
614                 then begin {MORE PARANIOD}
615                   work := op1 [ i ] + maxint                            589   587   589    29
```

```
616                 ; work := work + 1                                    589   589
617                 ; work := work - op2 [ i ]                            589   589   587   589
618                 ; work := work + maxint                               589   589    29
619                 ; tempresult [ i ] := work + 1                        591   589   589
620                 end
621                 else tempresult [ i ] := op1 [ i ] - op2 [ i ]        591   589   587   589   587   589
622               end
623               else begin
624                 borrowout := op1 [ i ] < op2 [ i ]                    590   587   589   587   589
625               ; tempresult [ i ] := op1 [ i ] - op2 [ i ]           591   589   587   589   587   589
626               end
627           ; if borrowin                                             590
628             then if tempresult [ i ] < - maxint                     591   589    29
629               then tempresult [ i ] := maxint                       591   589    29
630               else begin
631                 if tempresult [ i ] = 0                             591   589
632                 then borrowout := true                              590     S
633               ; tempresult [ i ] := pred ( tempresult [ i ] )       591   589     S   591   589
634               end
635           ; borrowin := borrowout                                   590   590
636           end
637         ; result := tempresult                                      587   591
638         end { MULTSUBTR }
639
640     ; function multge ( op1 , op2 : multint ) : boolean               *   587   587   291     S
641
642       ; var work : multint                                          589   291
643
644       ; begin
645           multsubtr ( op1 , op2 , work )                            586   640   640   642
646         ; multge := work [ multmct ] >= 0                           640   642   283
647         end { MULTGE }
648
649     ; function multgt ( op1 , op2 : multint ) : boolean               *   640   640   291     S
650
651       ; var work : multint                                          642   291
652
653       ; begin
654           multplint ( op2 , 1 , work )                              557   649   651
655         ; multsubtr ( op1 , work , work )                           586   649   651   651
656         ; multgt := work [ multmct ] >= 0                           649   651   283
657         end { MULTGT }
658
659 {% END MULTPRECISION}
660
661     ; procedure int2string ( fint : integer ; var fstring : string12 )    *     *    27     *   320
662
663       ; var lint , digit : integer                                      *     *    27
664
665       ; begin
666           lint := fint                                                663   661
667         ; i := 12                                                     589
668         ; fstring := '            '                                   661
669         ; if lint = 0                                                 663
670           then fstring [ 12 ] := '0'                                  661
671           else while ( lint <> 0 ) and ( i >= 1 )                     663   589
```

```
672             do begin
673               digit := lint mod 10                                  663   663
674             ; fstring [ i ] := chr ( ord ( '0' ) + digit )          661   589     S     S   663
675             ; i := pred ( i )                                       589     S   589
676             ; lint := lint div 10                                   663   663
677             end
678         end ( INT2STRING )
679
680     ; procedure reportstat ( fnum : integer ; fstr : string12 )       *     *    27     *   320
681
682       ; var lnumstr : string12                                        *   320
683         ; i : integer                                               589    27
684
685       ; begin
686           int2string ( fnum , lnumstr )                             661   680   682
687         ; for i := 1 to 12 do write ( summary , lnumstr [ i ] )     683   495   417   682   683
688         ; write ( summary , ' ' )                                   495   417
689         ; for i := 1 to 12 do write ( summary , fstr [ i ] )        683   495   417   680   683
690         ; writeln ( summary )                                       517   417
691         end ( REPORTSTAT )
692
693     ; procedure DisplayInteger ( Number : integer )                   *     *    27
694
695       ; var i : integer                                             683    27
696         ; lnumstr : string12                                        682   320
697
698       ; begin
699           int2string ( Number , lnumstr )                           661   693   696
700         ; i := 1                                                    695
701         ; while ( i <= 12 ) and ( lnumstr [ i ] = ' ' )             695   696   695
702           do i := succ ( i )                                        695     S   695
703         ; display ( ' ' )                                           142
704         ; while i <= 12                                             695
705           do begin
706             display ( lnumstr [ i ] )                               142   696   695
707           ; i := succ ( i )                                         695     S   695
708           end
709         ; display ( ' ' )                                           142
710         end (DisplayInteger )
711
712     ; procedure DisplayString ( s : StringType )                      *     *     *
713
714       ; var i : integer                                             695    27
715
716       ; begin
717           i := 1                                                    714
718         ; while ( s [ i ] <> '$' ) and ( i <= 100 )                 712   714   714
719           do begin
720             display ( s [ i ] )                                     142   712   714
721           ; i := i + 1                                              714   714
722           end
723         end (DisplayString )
724
725     ; procedure EmitError                                             *
726         ( errorCode : ErrorCodes ; sourceLineNo : integer )           *   255   349    27
727
```

```
728       ; var i : integer                                                714    27
729
730       ; begin
731           DisplayString ( 'epp $' )                                    712
732         ; case errorCode                                               726
733           of ePosStackOverflow                                         246
734             : DisplayString ( 'Position stack overflow $' )            712
735           ; ePosStackUnderflow                                         247
736             : DisplayString ( 'Position stack underflow $' )           712
737           ; eMarkStackOverflow                                         249
738             : DisplayString ( 'Mark stack overflow $' )                712
739           ; eMarkStackUnderflow                                        249
740             : DisplayString ( 'Mark stack underflow $' )               712
741           ; eCallStackOverflow                                         244
742             : DisplayString ( 'Call stack overflow $' )                712
743           ; eDuplicateSetup : DisplayString ( 'Token set up twice $' ) 250    712
744           ; eNotSetup : DisplayString ( 'Token not set up yet $' )     251    712
745           ; eSetupUnmatchedToken                                       252
746             : DisplayString ( 'Token not matched yet $' )              712
747           end {case }
748         ; DisplayString ( 'at line $' )                                712
749         ; DisplayInteger ( sourceLineNo )                              693    726
750         ; DisplayString ( 'S-code addr $' )                            712
751         ; DisplayInteger ( tablePointer )                              693    346
752         ; Display ( nl )                                                *     33
753         ; processing := false                                          342     S
754         ; gError := errorCode                                          375    726
755         ; i := 0                                                       728
756         ; i := 1 div i {ause abnormal termination}                     728    728
757       end {EmitError }
758
759     ; procedure getlrec ( var p : linerecptr )                          *     130    295
760
761       ; begin
762           if gfreep = nil                                              386
763           then begin new ( p ) ; linerecct := succ ( linerecct ) end    S     759    412     S    412
764           else begin p := gfreep ; gfreep := p ^ . dlink end           759    386    386    759    298
765         end { GETLREC }
766
767     ; procedure freelrec ( var p : linerecptr )                         *     759    295
768
769       ; begin
770           p ^ . dlink := gfreep                                        767    298    386
771         ; gfreep := p                                                  386    767
772         {% IF SINGLEPRECISION}
773         ; p ^ . tokno := 0                                             767    311
774         {% END SINGLEPRECISION}
775         {% IF MULTPRECISION}
776         ; multzero ( p ^ . tokno )                                     551    767    311
777         {% END MULTPRECISION}
778         end { FREELREC }
779
780     ; procedure unload                                                  *
781       ; forward                                                         *
782
783     ; procedure mark ( var m : markrec ; p : linerecptr )              134     *     311    767    295
```

```
784
785       ; begin
786           with m do begin linep := p ; tokno := p ^ . tokno end          783   311   783   311   783   311
787         end { MARK }
788
789     ; procedure setupcmnt                                                 *
790
791       ; begin
792           with gcurp ^                                                   385
793           do begin
794             length := succ ( length )                                    302     S   302
795           ; aggsize := gaggsize                                          299   393
796           ; if symbol = tcomment2                                        406     *
797             then begin
798               if startpos + length > rightmargin                         301   302   268
799               then startpos := rightmargin - length                      301   268   302
800             ; if gcurpos < startpos                                      390   301
801               then
802               else if gcurpos + 1 + length <= rightmargin                390   302   268
803                 then startpos := gcurpos + 1                             301   390
804               {% IF SINGLEPRECISION}
805             ; gaggsize := gaggsize + length                              393   393   302
806               {% END SINGLEPRECISION}
807               {% IF MULTPRECISION}
808             ; multplint ( gaggsize , length , gaggsize )                 557   393   302   393
809               {% END MULTPRECISION}
810             ; gcurpos := startpos + length                               390   301   302
811             ; newline := false                                           303     S
812             end
813             else if symbol = tcomment3                                   406     *
814               then begin
815                 startpos := gcurpos                                      301   390
816                  {% IF SINGLEPRECISION}
817               ; gaggsize := gaggsize + length                            393   393   302
818                  {% END SINGLEPRECISION}
819                  {% IF MULTPRECISION}
820               ; multplint ( gaggsize , length , gaggsize )               557   393   302   393
821                  {%  END MULTPRECISION}
822               ; newline := false                                         303     S
823               ; gcurpos := startpos + length                             390   301   302
824               end
825               else if symbol = tcomment4                                 406     *
826                 then begin
827                     {% IF SINGLEPRECISION}
828                   gaggsize                                              393
829                   := gaggsize                                           393
830                      + length                                           302
831                   {% END SINGLEPRECISION}
832                   {% IF MULTPRECISION}
833                      multplint ( gaggsize ,  length ,  gaggsize )       557   393   302   393
834                   {% END MULTPRECISION}
835                 ; newline := true                                        303     S
836                 ; gcurpos := startpos + length                           390   301   302
837                 end
838                 else if symbol = tcomment1                               406     *
839                   then begin
```

```
840                     newline := true                                 303     S
841                   {% IF SINGLEPRECISION}
842                   ; gaggsize := gaggsize + rightmargin - leftmargin 393   393   268   267
843                   {% END SINGLEPRECISION}
844                   {% IF MULTPRECISION}
845                   ; multplint                                       557
846                       ( gaggsize , rightmargin - leftmargin + 1     393   268   267
847                       , gaggsize )                                  393
848                   {% END MULTPRECISION}
849                   ; gcurpos := rightmargin                          390   268
850                   end
851           end
852         end { SETUPCMNT }
853
854       ; procedure skipcmnt                                          *
855
856         ; begin
857             with gcurp ^                                            385
858             do begin
859               aggsize := gaggsize                                   299   393
860               {% IF SINGLEPRECISION}
861             ; gaggsize := gaggsize + length                         393   393   302
862               {% END SINGLEPRECISION}
863               {% IF MULTPRECISION}
864             ; multplint ( gaggsize , length , gaggsize )            557   393   302   393
865               {% END MULTPRECISION}
866             ; if symbol = tcomment3                                 406     *
867               then startpos := gcurpos + 1                          301   390
868             ; if startpos + length > rightmargin                    301   302   268
869               then gcurpos := rightmargin                           390   268
870               else gcurpos := startpos + length                     390   301   302
871             end
872           end { SKIPCMNT }
873
874       ; procedure incharacter                                       *
875
876         ; begin
877             if not eof ( input )                                    487   417
878             then begin
879               if eoln ( input )                                     491   417
880               then begin
881                 pos := 1                                            378
882               ; sourceLineNo := succ ( sourceLineNo )               726     S   726
883               ; innewline := true                                   379     S
884               end
885               else pos := succ ( pos )                              378     S   378
886             ; get ( input )                                         422   417
887             end
888           end { INCHARACTER }
889
890       ; procedure insertchar ( fc : char )                          *     *     S
891
892         ; var p : linerecptr                                        783   295
893
894         ; begin
895             if i > fraglength                                       728   273
```

```
896           then begin
897             getirec ( p )                                          759   892
898           ; p ^ . rlink := newp ^ . rlink                          892   298   383   298
899           ; newp ^ . rlink := p                                    383   298   892
900           ; newp := p                                              383   892
901           ; i := 1                                                 728
902           end
903         ; newp ^ . text [ i ] := fc                                383   328   728   890
904         ; i := succ ( i )                                          728     S   728
905         ; newlength := succ ( newlength )                          377     S   377
906         end { INSERTCHAR }
907
908       ; procedure setupchar                                          *
909
910         ; begin
911             insertchar ( inputwindow ( input ) )                   890   539   417
912           ; incharacter                                            874
913           end { SETUPCHAR }
914
915       ; procedure prcomment                                          *
916
917         ; begin
918           while ( not eoln ( input ) )                             491   417
919                 and ( inputwindow ( input ) <> ';' )               539   417
920           do setupchar                                             908
921         ; if eoln ( input )                                        491   417
922           then cmntstate := comment                                381   381
923           else begin setupchar ; cmntstate := nocomment end        908   381   381
924         ; newsymbol := tcomment1                                   382     *
925         end { PRBRACECMNT }
926
927       ; function upperCase ( c : char ) : char                       *   495     S     S
928
929         ; begin
930           if ( ord ( 'a' {L.C.} ) <= ord ( c ) )                     S     S   927
931              and ( ord ( c ) <= ord ( 'z' {L.C.} ) )                 S   927     S
932           then upperCase                                           927
933             := chr ( ord ( c ) - ord ( 'a' {L.C.} ) + ord ( 'A' ) )  S     S   927     S     S
934           else upperCase := c                                      927   927
935         end
936
937       ; procedure lookupident                                        *
938
939         ; var lstring : string12                                     *   320
940           ; j , lo , hi , probe : integer                            *     *     *     *    27
941           ; exitloop : boolean                                       *     S
942
943         ; begin
944           if newlength > 12                                        377
945           then newsymbol := tident                                 382     *
946           else begin
947             lstring := '            '                              939
948           ; for j := 1 to newlength                                940   377
949             do lstring [ j ] := upperCase ( newp ^ . text [ j ] )  939   940   927   393   328   940
950           ; lo := rwptr [ newlength ] - 1                          940   401   377
951           ; hi := rwptr [ newlength + 1 ]                          940   401   377
```

```
 952          ; exitloop := false                                          941     S
 953          ; repeat if lo + 1 >= hi                                     940   940
 954              then begin newsymbol := tident ; exitloop := true end    382     *   941     S
 955              else begin
 956                probe := ( lo + hi ) div 2                             940   940   940
 957              ; if rw [ probe ] . string = lstring                     402   940   405   939
 958                then begin
 959                  newsymbol := rw [ probe ] . symbol                   382   402   940   406
 960                ; exitloop := true                                     941     S
 961                end
 962                else if rw [ probe ] . string < lstring                402   940   405   939
 963                  then lo := probe                                     940   940
 964                  else hi := probe                                     940   940
 965              end
 966            until exitloop                                           941
 967          ; if newsymbol = tcomment                                    382     *
 968            then begin setupchar ; prcomment end                       908   915
 969          end
 970        end { LOOKUPIDENT }
 971
 972    ; procedure number                                                   *
 973
 974      ; begin
 975          while inputwindow ( input ) in digits do setupchar           539   417   409   908
 976        ; if inputwindow ( input ) = '.' then setupchar                539   417   908
 977        ; while inputwindow ( input ) in digits do setupchar           539   417   409   908
 978        ; if inputwindow ( input ) = '&'                               539   417
 979          then begin
 980            setupchar                                                  908
 981          ; if (inputwindow ( input ) = '+') or                        539   417
 982            ( inputwindow ( input ) = '-')                             539   417
 983            then setupchar                                             908
 984          end
 985        ; while inputwindow ( input ) in digits do setupchar           539   417   409   908
 986        ; newsymbol := tnumber                                         382     *
 987        end { NUMBER }
 988
 989    {% IF PREP}
 990
 991    ; function prepcommand ( flp : linerecptr ) : boolean                *     *   295     S
 992
 993      ; begin
 994          {ASSERT: FLP POINTS TO A COMMENT}
 995          with flp ^ . rlink ^                                         991   298
 996          do if text [ 1 ] = '('                                       328
 997            then prepcommand := text [ 3 ] = '%'                       991   328
 998            else prepcommand := text [ 2 ] = '%'                       991   328
 999        end { PREPCOMMAND }
1000
1001    {% END PREP}
1002
1003    ; procedure insymbol ( skipping : boolean )                          *     *     S
1004
1005      ; var lsy : symboltyp                                              *   223
1006        ; laggsize : multint                                             *   291
1007        ; lnextp , llp1 , llp2 : linerecptr                              *     *     *   295
```

```
1008          ; lc : char                                                    *     S
1009          ; exitloop , string : boolean                                941   405     S
1010
1011        ; begin
1012            laggsize := gaggsize                                      1006   393
1013          ; if gcursy <> tendoffile                                    389     *
1014            then begin
1015              lnextp := gnextp                                        1007   385
1016            ; repeat getlrec ( newp )                                  759   383
1017              ; newp ^ . rlink := newp                                 383   298   383
1018              ; i := 1                                                 728
1019              ; newlength := 0                                         377
1020              ; if eoln ( input )                                      491   417
1021                then incharacter                                       874
1022              ; newstartpos := pos                                     378   378
1023              ; if cmntstate = comment                                 381   381
1024                then prcomment                                         915
1025                else repeat exitloop := true                          1009     S
1026                  ; newstartpos := pos                                 378   378
1027                  ; if eof ( input )                                   487   417
1028                    then newsymbol := tendoffile                       382     *
1029                    else if eoln ( input )                             491   417
1030                      then begin incharacter ; exitloop := false end   874  1009     S
1031                      else if inputwindow ( input ) in lexchars        539   417   409
1032                        then case inputwindow ( input )                539   417
1033                          of 'A' , 'B' , 'C' , 'D' , 'E' , 'F' , 'G'
1034                           , 'H' , 'I' , 'J' , 'K' , 'L' , 'M' , 'N'
1035                           , 'O' , 'P' , 'Q' , 'R' , 'S' , 'T' , 'U'
1036                           , 'V' , 'W' , 'X' , 'Y' , 'Z' , 'a' , 'b'
1037                           , 'c' , 'd' , 'e' , 'f' , 'g' , 'h' , 'i'
1038                           , 'j' , 'k' , 'l' , 'm' , 'n' , 'o' , 'p'
1039                           , 'q' , 'r' , 's' , 't' , 'u' , 'v' , 'w'
1040                           , 'x' , 'y' , 'z'
1041                           : begin
1042                             while inputwindow ( input )               539   417
1043                                   in letters + digits + [ '_' ]       409   409
1044                             do setupchar                              908
1045                           ; lookupident                               937
1046                           end
1047                         ; '"'
1048                           : begin
1049                             setupchar                                 908
1050                           ; if inputwindow ( input ) <> '"'           539   417
1051                             then string := true                      1009     S
1052                             else begin string := false ; setupchar end 1009     S   908
1053                           ; if not eof ( input )                      487   417
1054                                and not eoln ( input )                 491   417
1055                             then while string                        1009
1056                                       and ( not eof ( input ) )       487   417
1057                                       and ( not eoln ( input ) )      491   417
1058                               do begin
1059                                 setupchar                             908
1060                               ; if inputwindow ( input ) = '"'        539   417
1061                                 then begin
1062                                   setupchar                           908
1063                                 ; if inputwindow ( input ) <> '"'     539   417
```

```
1064                                then string := false                    1009     S
1065                              end
1066                            end
1067                        ; newsymbol := tsimplestring                    382      *
1068                        end
1069                      ; '''
1070                        : begin
1071                          setupchar                                     908
1072                        ; repeat setupchar                              908
1073                          until ( inputwindow ( input ) = ''' )         539    417
1074                                or ( eof ( input ) )                    487    417
1075                                or ( eoln ( input ) )                   491    417
1076                        ; if inputwindow ( input ) = '''                539    417
1077                          then setupchar                                908
1078                        ; newsymbol := tcharsequence                    382      *
1079                        end
1080                      ; '&'
1081                        : begin
1082                          setupchar                                     908
1083                        ; newsymbol := tbasisOfTen                      382      *
1084                        end
1085                      ; '0' , '1' , '2' , '3' , '4' , '5' , '6'
1086                        , '7' , '8' , '9'
1087                        : number                                        972
1088                      ; '<'
1089                        : begin
1090                          setupchar                                     908
1091                        ; if inputwindow ( input ) = '='                539    417
1092                          then begin
1093                            setupchar                                   908
1094                          ; newsymbol := tle                            382      *
1095                          end
1096                          else if inputwindow ( input ) = '>'           539    417
1097                            then begin
1098                              setupchar                                 908
1099                            ; newsymbol := tne                          382      *
1100                            end
1101                            else newsymbol := tlt                       382      *
1102                        end
1103                      ; '>'
1104                        : begin
1105                          setupchar                                     908
1106                        ; if inputwindow ( input ) = '='                539    417
1107                          then begin
1108                            setupchar                                   908
1109                          ; newsymbol := tge                            382      *
1110                          end
1111                          else newsymbol := tgt                         382      *
1112                        end
1113                      ; ':'
1114                        : begin
1115                          setupchar                                     908
1116                        ; if inputwindow ( input ) = '='                539    417
1117                          then begin
1118                            setupchar                                   908
1119                          ; newsymbol := tbecomes                       382      *
```

```
1120                          end
1121                          else if inputwindow ( input ) = '-'          539   417
1122                            then begin
1123                              setupchar                                908
1124                            ; newsymbol := tdenotes                    382     *
1125                            end
1126                            else newsymbol := tcolon                   382     *
1127                        end
1128                      ; '*'
1129                        : begin
1130                          setupchar                                    908
1131                        ; if inputwindow ( input ) = '*'               539   417
1132                          then begin
1133                            setupchar                                  908
1134                          ; newsymbol := texpon                        382     *
1135                          end
1136                          else newsymbol := ttimes                     382     *
1137                        end
1138                      ; '/'
1139                        : begin
1140                          setupchar                                    908
1141                        ; if inputwindow ( input ) = '/'               539   417
1142                          then setupchar                               908
1143                        ; newsymbol := tdivide                         382     *
1144                        end
1145                      ; '='
1146                        : begin
1147                          setupchar                                    908
1148                        ; if inputwindow ( input ) = '='               539   417
1149                          then begin
1150                            setupchar                                  908
1151                          ; newsymbol := trefEqual                     382     *
1152                          end
1153                          else if inputwindow ( input ) = '/'          539   417
1154                            then begin
1155                              setupchar                                908
1156                            ; if inputwindow ( input ) = '='           539   417
1157                              then begin
1158                                setupchar                              908
1159                              ; newsymbol := trefNotEqual              382     *
1160                              end
1161                            end
1162                          else newsymbol := teq                        382     *
1163                        end
1164                      ; '!' : begin setupchar ; prcomment end          908   915
1165                      ; '(' , ')' , '[' , ']' , ',' , ';' , '.'
1166                        , '+' , '-'
1167                        : begin
1168                          case inputwindow ( input )                   539   417
1169                          of '(' , '[' : lsy := tleftParen            1005     *
1170                          ; ')' , ']' : lsy := trightParen            1005     *
1171                          ; ',' : lsy := tcomma                       1005     *
1172                          ; ';' : lsy := tsemicolon                   1005     *
1173                          ; '.' : lsy := tdot                         1005     *
1174                          ; '+' : lsy := tplus                        1005     *
1175                          ; '-' : lsy := tminus                       1005     *
```

```
1176                          end
1177                        ; setupchar                                    908
1178                        ; newsymbol := lsy                             382  1005
1179                        end
1180                      ; ' '
1181                        : begin
1182                          incharacter                                  874
1183                        ; exitloop := false                            1009    S
1184                        end
1185                      end { CASE }
1186                    else   { NOT INPUTWINDOW ( INPUT ) IN LEXCHARS }
1187                    begin
1188                      while ( not eoln ( input ) )                     491   417
1189                            and not ( inputwindow ( input )            539   417
1190                                      in lexchars                      409
1191                                    )
1192                      do setupchar                                     908
1193                    ; newsymbol := tunknown                            382    *
1194                    end
1195                until exitloop                                       1009
1196              ; insertchar ( ' ' )                                     890
1197              ; with newp ^                                            383
1198                do begin
1199                  length := pred ( newlength )                       302    S   377
1200                ; symbol := newsymbol                                406  382
1201                ; startpos := newstartpos                            301  378
1202                ; newline := innewline                               303  379
1203                ; dlink := nil                                       298
1204                {% IF SINGLEPRECISION}
1205                ; gtokno := succ ( gtokno )                          392    S   392
1206                {% END SINGLEPRECISION}
1207                {% IF MULTPRECISION}
1208                ; multplint ( gtokno , 1 , gtokno )                  557  392  392
1209                {% END MULTPRECISION}
1210                ; tokno := gtokno                                    311  392
1211                ; sepafter                                           305
1212                  := inputwindow ( input ) in [ ' ' ] + throwaways    539  417  410
1213                end
1214              ; lnextp ^ . dlink := newp                             1007  298  383
1215              ; lnextp := newp                                       1007  383
1216              ; innewline := false                                   379    S
1217              until not ( newsymbol                                  382
1218                          in [ tcomment1 , tcomment2 , tcomment3      *     *    *
1219                             , tcomment4 ]                            *
1220                        )
1221            ; tokenct := succ ( tokenct )                            412    S   412
1222            ; gcompp := gcurp                                        385  385
1223            ; while gcurp <> gnextp                                  385  385
1224              do begin
1225                if gcurp ^ . symbol                                  385  406
1226                   in [ tcomment1 , tcomment2 , tcomment3 , tcomment4 ]  *    *    *    *
1227                then if skipping then skipcmnt else setupcmnt        1003  854  789
1228              ; gtrailp := gcurp                                     385  385
1229              ; gcurp := gcurp ^ . dlink                             385  385  298
1230              ; if not ( gcompp ^ . symbol in [ tcomment1 , tcomment4 ]  385  406   *    *
1231                       )
```

```
1232              then begin gcomptrailp := gtrailp ; gcompp := gcurp end
1233            end
1234          ; gcurp ^ . aggsize := gaggsize
1235          ; gnextp := lnextp
1236          ; gcursy := gcurp ^ . symbol
1237          ; llp1 := gcurp
1238          ; llp2 := llp1 ^ . dlink
1239          ; while llp2 <> gnextp
1240            do begin
1241              with llp2 ^
1242              do begin
1243                if newline
1244                then if dlink ^ . newline
1245                  then symbol := tcomment1
1246                  else symbol := tcomment4
1247                else if llp1 ^ . startpos + llp1 ^ . length + 1
1248                        >= startpos
1249                  then symbol := tcomment3
1250                  else symbol := tcomment2
1251              {% IF PREP}
1252              ; moveable := not prepcommand ( llp2 )
1253              {% END PREP}
1254              {% IF NOT PREP}
1255              ; moveable := true
1256              {% END NOT PREP}
1257              end
1258            ; llp1 := llp2
1259            ; llp2 := llp2 ^ . dlink
1260            end
1261        { ; IF GAGGSIZE - LAGGSIZE > RIGHTMARGIN - LEFTMARGIN
1262            THEN ** UNLOAD ** }
1263          end { IF GCURSY <> SYEOF }
1264        ; setupState := unmatched
1265        end { INSYMBOL }
1266
1267    ; procedure blankline
1268
1269      ; var p : linerecptr
1270        ; ltokno , laggsize : multint
1271
1272      ; begin
1273          if ( gtrailp = nil ) or ( gtrailp ^ . symbol <> tblankline )
1274          then begin
1275            getlrec ( p )
1276        {% IF SINGLEPRECISION}
1277          ; gtokno := succ ( gtokno )
1278        {% END SINGLEPRECISION}
1279        {% IF MULTPRECISION}
1280          ; multplint ( gtokno , 1 , gtokno )
1281        {% END MULTPRECISION}
1282          ; ltokno := gcurp ^ . tokno
1283          ; with p ^
1284            do begin
1285              symbol := tblankline
1286            ; rlink := p
1287            ; dlink := gcurp
```

385	385	385	385	
385	299	393		
385	1007			
389	385	406		
1007	385			
1007	1007	298		
1007	385			
1007				
303				
298	303			
406	*			
406	*			
1007	301	1007	302	
301				
406	*			
406	*			
306	991	1007		
306	S			
1007	1007			
1007	1007	298		
388	388			
*				
892	295			
*	1006	291		
385	385	406	*	
759	1269			
392	S	392		
557	392	392		
1270	385	311		
1269				
406	*			
298	1269			
293	385			

```
1288            ; length := rightmargin - leftmargin + 1                    302   268   267
1289            end
1290          ; laggsize := gcurp ^ . aggsize                              1270   385   299
1291          ; if gfirstp = gcurp                                          385   385
1292            then gfirstp := p                                           385  1269
1293            else gtrailp ^ . dlink := p                                 385   298  1269
1294          ; gtrailp := p                                                385  1269
1295        {% IF SINGLEPRECISION}
1296          ; gaggsize := gaggsize + rightmargin - leftmargin + 1         393   393   268   267
1297        {% END SINGLEPRECISION}
1298        {% IF MULTPRECISION}
1299          ; multplint                                                   557
1300              ( gaggsize , rightmargin - leftmargin + 1 , gaggsize )    393   268   267   393
1301        {% END MULTPRECISION}
1302          ; if gcompp = gcurp                                           385   385
1303            then begin
1304              gcompp := p                                               385  1269
1305            ; gcomptrailp := nil { NOT THE RIGHT VALUE FOR              385
c 1306                                 GCOMPTRAILP, BUT IT WOULD
c 1307                                 TAKE A LOOP TO FIND, AND
c 1308                                 WE CANNOT USE IT ANYWAY }
1309            end
1310            else if gcompp ^ . symbol <> tblankline                     385   406     *
1311              then begin
1312                getlrec ( p )                                           759  1269
1313              {% IF SINGLEPRECISION}
1314                ; gtokno := succ ( gtokno )                             392     S   392
1315              {% END SINGLEPRECISION}
1316              {% IF MULTPRECISION}
1317                ; multplint ( gtokno , 1 , gtokno )                     557   392   392
1318              {% END MULTPRECISION}
1319                ; ltokno := gcompp ^ . tokno                           1270   385   311
1320                ; with p ^                                             1269
1321                  do begin
1322                    symbol := tblankline                                406     *
1323                  ; rlink := p                                          298  1269
1324                  ; dlink := gcompp                                     298   385
1325                  ; length := rightmargin - leftmargin + 1              302   268   267
1326                  end
1327                ; laggsize := gcompp ^ . aggsize                       1270   385   299
1328                ; if gfirstp = gcompp                                   385   385
1329                  then begin
1330                    gfirstp := p                                        335  1269
1331                  ; gcomptrailp := p                                    385  1269
1332                  ; gtrailp := p                                        385  1269
1333                  end
1334                  else gcomptrailp ^ . dlink := p                       335   298  1269
1335              {% IF SINGLEPRECISION}
1336                ; gaggsize := gaggsize + rightmargin - leftm rgin + 1  393   393   268   267
1337              {% END SINGLEPRECISION}
1338              {% IF MULTPRECISION}
1339                ; multplint                                             557
1340                    ( gaggsize , rightmargin - leftmargin + 1 , gaggsize 393   268   267   393
1341                    )
1342              {% END MULTPRECISION}
1343                ; gcompp := p                                           385  1269
```

```
1344              end
1345            ; while p <> nil                                          1269
1346              do with p ^                                             1269
1347                do begin
1348                  tokno := ltokno                                      311  1270
1349              {% IF SINGLEPRECISION}
1350                ; ltokno := succ ( ltokno )                           1270     S  1270
1351              {% END SINGLEPRECISION}
1352              {% IF MULTPRECISION}
1353                ; multplint ( ltokno , 1 , ltokno )                   557  1270  1270
1354              {% END MULTPRECISION}
1355                ; aggsize := laggsize                                 299  1270
1356                {% IF SINGLEPRECISION}
1357                ; laggsize := laggsize + length                      1270  1270   302
1358                {% END SINGLEPRECISION}
1359                {% IF MULTPRECISION}
1360                ; multplint ( laggsize , length , laggsize )          557  1270   302  1270
1361                {% END MULTPRECISION}
1362                ; p := dlink                                         1269   298
1363                end
1364          end {if (gtrailp = nil) or ( ... ) }
1365        end { BLANKLINE }
1366
1367      ; procedure unload                                               780
1368
1369        ; var lpos : positiontyp                                         *   286
1370          ; llp1 , llp2 , llp3 , llp4 : linerecptr                    1007  1007     *     *   295
1371          ; i : integer                                                728    27
1372          {% IF MULTPRECISION}
1373          ; lmult : multint                                              *   291
1374
1375          {% END MULTPRECISION}
1376
1377        ; begin
1378          {% IF MULTPRECISION}
1379            multsubtr                                                  586
1380              ( gcompp ^ . aggsize , gfirstp ^ . aggsize , lmult )     385   299   385   299  1373
1381          ; if ( gfirstp <> gcompp )                                   385   385
1382               and ( multgt ( lmult , linespace )                      649  1373   399
1383                     or ( gcompp ^ . symbol = tendoffile )             385   406     *
1384                  )
1385            then
1386          {% END MULTPRECISION}
1387          {% IF SINGLEPRECISION}
1388              if ( gfirstp <> gcompp )                                 385   385
1389                 and ( ( gcompp ^ . aggsize - gfirstp ^ . aggsize      385   299   385   299
1390                         > rightmargin - leftmargin + 1                268   267
1391                       )
1392                       or ( gcompp ^ . symbol = tendoffile )           385   406     *
1393                    )
1394              then
1395          {% END SINGLEPRECISION}
1396              begin
1397                llp1 := gfirstp                                      1370   385
1398              ; while llp1 <> gcompp                                 1370   385
1399                do begin
```

```
1400                if llp1 ^ . newline                                    1370   303
1401                   or ( llp1 ^ . startpos < outpos )                   1370   301   390
1402                   or ( llp1 ^ . symbol = tblankline )                 1370   406    *
1403                then begin
1404                  writeln ( output )                                    517   417
1405                ; outpos := 1                                           390
1406                ; nltokno := llp1 ^ . tokno                             392  1370   311
1407                ; lineoutct := succ ( lineoutct )                       412     S   412
1408                end
1409              ; if llp1 ^ . symbol = tblankline                         1370   406    *
1410                then outpos := rightmargin                              390   268
1411                else begin
1412                  while outpos < llp1 ^ . startpos                      390  1370   301
1413                  do begin
1414                    write ( output , ' ' )                              495   417
1415                  ; outpos := succ ( outpos )                           390     S   390
1416                  end
1417                ; llp2 := llp1 ^ . rlink                               1370  1370   298
1418                ; i := 1                                               1371
1419                ; for lpos := 1 to llp1 ^ . length                     1369  1370   302
1420                  do begin
1421                    if i > fraglength                                  1371   273
1422                    then begin llp2 := llp2 ^ . rlink ; i := 1 end     1370  1370   298  1371
1423                  ; write ( output , llp2 ^ . text [ i ] )              495   417  1370   328  1371
1424                  ; i := succ ( i )                                    1371     S  1371
1425                  end
1426                ; outpos := outpos + llp1 ^ . length                    390   390  1370   302
1427                end
1428              ; outtokno := llp1 ^ . tokno                              392  1370   311
1429              ; llp2 := llp1                                           1370  1370
1430              ; llp1 := llp1 ^ . dlink                                 1370  1370   298
1431              ; llp4 := llp2                                           1370  1370
1432              ; repeat llp3 := llp2                                    1370  1370
1433                ; llp2 := llp2 ^ . rlink                               1370  1370   298
1434                ; freelrec ( llp3 )                                     767  1370
1435                until llp2 = llp4                                      1370  1370
1436              end { WHILE }
1437            ; gfirstp := llp1                                           385  1370
1438            ; if gcompp = gcurp                                         385   385
1439              then gtrailp := nil                                       385
1440            ; gcomptrailp := nil                                        385
1441            end { IF GFIRSTP <> GCOMPP ETC. }
1442        end { UNLOAD }
1443
1444    ; procedure PushConstPos ( element : positiontyp )                    *     *   286
1445
1446      ; begin
1447          if PosStackTop < MaxPosStack                                  396   269
1448          then begin
1449            PosStackTop := PosStackTop + 1                              396   396
1450          ; PosStack [ PosStackTop ] := element                         394   396  1444
1451          end
1452          else begin
1453            EmitError ( ePosStackOverflow , sourceLineNo )              725   246   726
1454          ; processing := false                                         342     S
1455          end
```

```
1456          end  {  PushConstPos  }                                                    
1457                                                                                     
1458        ; procedure PushPos ( element : Positiontyp )                          *  1444     *
1459                                                                                     
1460          ; begin                                                                    
1461              if PosStackTop < MaxPosStack                                    396   269      
1462              then begin                                                             
1463                PosStackTop := PosStackTop + 1                                396   396      
1464              ; PosStack [ PosStackTop ] := element                           394   396  1458
1465              end                                                                    
1466              else begin                                                             
1467                EmitError ( ePosStackOverflow , sourceLineNo )                725   246   726
1468              ; processing := false                                           342     S      
1469              end                                                                    
1470            end  {  PushPos  }                                                       
1471                                                                                     
1472        ; procedure PushMark ( m : markrec )                                   *  783   311
1473                                                                                     
1474          ; begin                                                                    
1475              if MarkStackTop < MaxMarkStack                                  397   270      
1476              then begin                                                             
1477                MarkStackTop := MarkStackTop + 1                              397   397      
1478              ; MarkStack [ MarkStackTop ] := m                               395   397  1472
1479              end                                                                    
1480              else begin                                                             
1481                EmitError ( eMarkStackOverflow , sourceLineNo )               725   248   726
1482              ; processing := false                                           342     S      
1483              end                                                                    
1484            end  {  PushMark  }                                                      
1485                                                                                     
1486        ; procedure PopPos                                                     *               
1487                                                                                     
1488          ; begin                                                                    
1489              if PosStackTop > 0                                              396               
1490              then PosStackTop := PosStackTop - 1                             396   396         
1491              else begin                                                             
1492                EmitError ( ePosStackUnderflow , sourceLineNo )               725   247   726
1493              ; processing := false                                           342     S      
1494              end                                                                    
1495            end  {  PopPos  }                                                        
1496                                                                                     
1497        ; procedure PopMark                                                    *               
1498                                                                                     
1499          ; begin                                                                    
1500              if MarkStackTop > 0                                             397               
1501              then MarkStackTop := MarkStackTop - 1                           397   397         
1502              else begin                                                             
1503                EmitError ( eMarkStackUnderflow , sourceLineNo )              725   249   726
1504              ; processing := false                                           342     S      
1505              end                                                                    
1506            end  {  PopMark  }                                                       
1507                                                                                     
1508        ; procedure SwapMarks                                                  *               
1509                                                                                     
1510          ; var lMark : markrec                                                *   311         
1511                                                                                     
```

```
1512        ; begin
1513            lMark := MarkStack [ MarkStackTop ]                                  1510   395   397
1514          ; MarkStack [ MarkStackTop ] := MarkStack [ MarkStackTop - 1 ]          395   397   395   397
1515          ; MarkStack [ MarkStackTop - 1 ] := lMark                              395   397  1510
1516          end {SwapMarks }
1517
1518      ; procedure setup ( fpos : positiontyp ; fnewline : boolean )               *     *    286     *     S
1519
1520        ; var llength , lstart : integer                                          *     *     27
1521
1522        ; begin
1523            case setupState                                                      388
1524            of unmatched                                                        388
1525              : EmitError ( eSetupUnmatchedToken , sourceLineNo )               725   252   726
1526            ; matchFailed : setupState := unmatched                             388   388   388
1527            ; matched                                                           388
1528              : with gcurp ^                                                    385
1529                do begin
1530                  length := succ ( length )                                     302     S   302
1531                ; if dlink ^ . symbol = tcomment3                               298   406     *
1532                  then llength := length + dlink ^ . length                    1520   302   298   302
1533                  else llength := length                                       1520   302
1534                ; if fpos + llength > rightmargin                              1518  1520   268
1535                  then begin
1536                    lstart := rightmargin - llength                            1520   268  1520
1537                  ; if lstart < leftmargin                                     1520   267
1538                    then startpos := leftmargin                                 301   267
1539                    else startpos := lstart                                     301  1520
1540                  end
1541                  else startpos := fpos                                         301  1518
1542                ; newline := fnewline                                           303  1518
1543                ; aggsize := gaggsize                                          299   393
1544            {% IF SINGLEPRECISION}
1545                ; gaggsize := gaggsize + length                                393   393   302
1546            {% END SINGLEPRECISION}
1547            {% IF MULTPRECISION}
1548                ; multplint ( gaggsize , length , gaggsize )                   557   393   302   393
1549            {% END MULTPRECISION}
1550                ; gcurpos := startpos + length                                 390   301   302
1551          { ASSERT: NOT ( SYMBOL
c 1552                        IN [ TCOMMENT1 , TCOMMENT2 , TCOMMENT3 , TCOMMENT4 ]
c 1553                        ) }
1554                ; gcurposnc := gcurpos                                         390   390
1555                ; skip := false                                                304     S
1556                ; setupState := unmatched                                      388   388
1557                ; insymbol ( false )                                          1003     S
1558                end {with }
1559            end {case }
1560          end { SETUP }
1561
1562      ; procedure ccompress ( m : markrec )                                       *  1472   311
1563
1564        ; var lpos : positiontyp                                               1369   286
1565          ; llp : linerecptr                                                      *   295
1566          {% IF MULTPRECISION}
1567          ; lmult : multint                                                    1373   291
```

```
1568
1569          {% END MULTPRECISION}
1570
1571        ; begin
1572          {% IF MULTPRECISION}
1573            multplint ( m . tokno , 1 , lmult )                          557  1562   311  1567
1574          ; if multgt ( gcompp ^ . tokno , lmult )                       649   395   311  1567
1575               and multge ( m . tokno , nltokno )                        640  1562   311   392
1576            then
1577          {% END MULTPRECISION}
1578          {% IF SINGLEPRECISION}
1579              if ( m . tokno + 1 < gcompp ^ . tokno )                    1562   311   385   311
1580                 and ( m . tokno >= nltokno )                            1562   311   392
1581              then
1582          {% END SINGLEPRECISION}
1583              begin
1584            {% IF SINGLEPRECISION}
1585                if m . tokno > outtokno                                  1562   311   392
1586                then
1587            {% END SINGLEPRECISION}
1588            {% IF MULTPRECISION}
1589                  if multgt ( m . tokno , outtokno )                      649  1562   311   392
1590                  then
1591            {% END MULTPRECISION}
1592                  begin
1593                    lpos                                                1564
1594                    := m . linep ^ . startpos + m . linep ^ . length     1562   311   301  1562   311   302
1595                  ; llp := m . linep ^ . dlink                           1565  1562   311   298
1596                  end
1597                  else begin llp := gfirstp ; lpos := outpos end         1565   385  1564   390
1598          {% IF MULTPRECISION}
1599              ; multsubtr                                                586
1600                  ( gcompp ^ . aggsize , llp ^ . aggsize , lmult )       385   299  1565   299  1567
1601              ; multplint ( lmult , lpos , lmult )                       557  1567  1564  1567
1602              ; if multge ( multrightmargin , lmult )                    640   399  1567
1603                then if multgt ( gcompp ^ . tokno , llp ^ . tokno ) then 649   385   311  1565   311
1604          {% END MULTPPECISION}
1605              {% IF SINGLEPRECISION}
1606              ; if ( lpos + gcompp ^ . aggsize - llp ^ . aggsize )       1564   385   299  1565   299
1607                   <= rightmargin                                        268
1608                then if llp ^ . tokno < gcompp ^ . tokno                 1565   311   385   311
1609                then
1610              {% END SINGLEPRECISION}
1611                  begin
1612                    while llp <> gcompp                                  1565   385
1613                    do begin
1614                      if llp ^ . symbol in [ tcomment2 , tcomment4 ]     1565   406     *     *
1615                      then lpos := llp ^ . dlink ^ . startpos            1564  1565   298   301
1616                      else begin
1617                        llp ^ . startpos := lpos                         1565   301  1564
1618                      ; lpos := lpos + llp ^ . length                    1564  1564  1565   302
1619                      ; if llp ^ . skip                                  1565   304
1620                        then lpos := succ ( lpos )                       1564     S  1564
1621                      ; llp ^ . newline := false                         1565   303     S
1622                      end
1623                    ; llp := llp ^ . dlink                               1565  1565   298
```

```
1624                    end
1625                  ; if gcompp = gcurp                                  385   385
1626                    then gcurpos := lpos                               390  1564
1627                  ; gcurposnc := lpos                                  390  1564
1628                  end
1629                  else
1630                else unload                                           1367
1631              end
1632          end { CCOMPRESS }
1633
1634      ; procedure movecmnts                                             *
1635
1636        ; var lcurp : linerecptr                                        *   295
1637          ; laggsize , ltokno : multint                                1270  1270   291
1638        {% IF MULTPRECISION}
1639          ; lmult : multint                                            1567   291
1640
1641        {% END MULTPRECISION}
1642
1643        ; begin
1644            laggsize := gaggsize                                       1637   393
1645          ; lcurp := gcurp                                             1636   385
1646          ; if ( gcurp ^ . dlink <> gnextp )                            385   298   385
1647               and ( gcurp ^ . dlink ^ . moveable )  ,                  385   298   306
1648            then begin
1649              gcurp := gcurp ^ . dlink                                  385   385   298
1650            ; if gfirstp = lcurp                                        385  1636
1651              then gfirstp := gcurp                                     385   385
1652              else gtrailp ^ . dlink := gcurp                           385   298   385
1653            ; if gcompp = lcurp                                         385  1636
1654              then gcompp := gcurp                                      385   385
1655            ; ltokno := lcurp ^ . tokno                                1637  1636   311
1656            ; while ( gcurp <> gnextp ) and ( gcurp ^ . moveable )       385   385   385   306
1657              do begin
1658                gcurp ^ . tokno := ltokno                               385   311  1637
1659              {% IF SINGLEPRECISION}
1660                ; ltokno := succ ( ltokno )                            1637     5  1637
1661              {% END SINGLEPRECISION}
1662              {% IF MULTPRECISION}
1663                ; multplint ( ltokno , 1 , ltokno )                     557  1637  1637
1664              {% END MULTPRECISION}
1665                ; setupcmnt                                             789
1666                ; gtrailp := gcurp                                      385   385
1667                ; gcurp := gcurp ^ . dlink                              385   385   298
1668                ; if not ( gcompp ^ . symbol in [ tcomment1 , tcomment4 ]  385   406     *     *
1669                       )
1670                  then begin gcomptrailp := gtrailp ; gcompp := gcurp end   385   385   385   385
1671              end
1672            ; lcurp ^ . tokno := ltokno                                1636   311  1637
1673            ; lcurp ^ . dlink := gcurp                                 1636   298   385
1674            ; gtrailp ^ . dlink := lcurp                                385   298  1636
1675            ; if gcompp = gcurp                                         385   385
1676              then gcompp := lcurp                                      385  1636
1677            ; gcurp := lcurp                                            385  1636
1678            ; gcurp ^ . aggsize := gaggsize                             385   299   393
1679          {% IF SINGLEPRECISION}
```

```
1630          ; if gaggsize - laggsize > rightmargin - leftmargin          393  1637   268   267
1681            then unload                                               1367
1682        {% END SINGLEPRECISION}
1683        {% IF MULTPRECISION}
1684          ; multsubtr ( gaggsize , laggsize , lmult )                  586   393  1637  1639
1685          ; if multge ( lmult , linespace )                            640  1639   399
1686            then unload                                               1367
1687        {% END MULTPRECISION}
1688          end
1689        end { MOVECMNTS }
1690
1691    ; procedure skip ( fsys : setofsymbol ; displaySet : boolean )      304     *   224     *     S
1692
1693      ; var laggsize : multint                                         1637   291
1694        ; i : integer                                                  1371    27
1695      {% IF MULTPRECISION}
1696        ; lmult : multint                                              1639   291
1697
1698      {% END MULTPRECISION}
1699
1700      ; begin
1701          laggsize := gaggsize                                         1693   393
1702        ; if displaySet                                                1691
1703          then begin
1704          ; DisplayString ( 'skipSet$' )                               712
1705          ; for i := 0 to 127                                          1694
1706            do if i in fsys                                            1694  1691
1707              then DisplayInteger ( i )                                693  1694
1708          ; display ( nl )                                             142    33
1709          end
1710        ; while not ( gcursy in fsys )                                 389  1691
1711          do begin
1712            with gcurp ^                                               385
1713            do begin
1714              skip := true                                            1691     S
1715            ; if displaySet                                           1691
1716              then begin
1717              ; DisplayString ( 'skipSymbol$' )                        712
1718              ; DisplayInteger ( gcursy )                              693   389
1719              ; DisplayInteger ( tokno )                               693   311
1720              ; display ( nl )                                         142    33
1721              end
1722            ; aggsize := gaggsize                                      299   393
1723            ; if ( dlink ^ . symbol in fsys ) and sepafter             298   406  1691   305
1724              then length := succ ( length )                           302     S   302
1725          {% IF SINGLEPRECISION}
1726            ; gaggsize := gaggsize + length                            393   393   302
1727          {% END SINGLEPRECISION}
1728          {% IF MULTPRECISION}
1729            ; multplint ( gaggsize , length , gaggsize )               557   393   302   393
1730          {% END MULTPRECISION}
1731            ; skipct := succ ( skipct )                                412     S   412
1732            ; if startpos + length > rightmargin                       301   302   268
1733              then gcurpos := rightmargin                              390   268
1734              else gcurpos := startpos + length                        390   301   302
1735            ; gcurposnc := gcurpos                                     390   390
```

```
1736          {% IF SINGLEPRECISION}
1737            ; if gaggsize - laggsize > rightmargin - leftmargin + 1       393  1693   268   267
1738              then begin unload ; laggsize := gaggsize end               1367  1693   393
1739          {% END SINGLEPRECISION}
1740          {% IF MULTPRECISION}
1741            ; multsubtr ( gaggsize , laggsize , lmult )                   586   393  1693  1696
1742            ; if multgt ( lmult , linespace )                             649  1696   399
1743              then begin unload ; laggsize := gaggsize end               1367  1693   393
1744          {% END MULTPRECISION}
1745            end
1746          ; insymbol ( true )                                            1003     S
1747          end
1748        ; if displaySet                                                  1691
1749          then begin
1750            DisplayString ( 'ResumeLine$' )                               712
1751          ; DisplayInteger ( sourceLineNo )                               693   726
1752          ; DisplayString ( 'token$' )                                    712
1753          ; DisplayInteger ( gcursy )                                     693   389
1754          ; DisplayInteger ( gcurp ^ . tokno )                            693   385   311
1755          ; display ( nl )                                                142    33
1756          end
1757        end { SKIP }
1758
1759    ; function indent ( fpos : positiontyp ) : positiontyp                  *  1518   286   286
1760
1761      ; var i : integer                                                  1694    27
1762
1763      ; begin
1764          i := fpos + indentamt                                          1761  1759   272
1765        ; if i <= rightmargin                                            1761   268
1766          then indent := i                                               1759  1761
1767          else indent := rightmargin                                     1759   268
1768        end { INDENT }
1769
1770    ; procedure initrw ( fsymbol : symboltyp ; fstring : string12 )          *     *   223   661   320
1771
1772      ; var len , i , j : integer                                            *  1761   940    27
1773
1774      ; begin
1775          if rwct < rwmax - 1                                             408   274
1776          then begin
1777            len := 12                                                    1772
1778          ; while ( len >= 2 ) and ( fstring [ len ] = ' ' )             1772  1770  1772
1779            do len := pred ( len )                                       1772     S  1772
1780          ; i := rwptr [ len ]                                           1772   401  1772
1781          ; j := rwptr [ len + 1 ]                                       1772   401  1772
1782          ; while ( i < j ) and ( rw [ i ] . string < fstring )          1772  1772   402  1772  1009  1770
1783            do i := succ ( i )                                           1772     S  1772
1784          ; if rw [ i ] . string <> fstring                               402  1772  1009  1770
1785            then begin
1786              j := rwptr [ 13 ] - 1                                      1772   401
1787            ; while j >= i                                               1772  1772
1788              do begin
1789                rw [ j + 1 ] := rw [ j ]                                  402  1772   402  1772
1790              ; j := pred ( j )                                          1772     S  1772
1791              end
```

```
1792            ; rw [ i ] . string := fstring                          402  1772  1009  1770
1793            ; for j := len + 1 to 13                               1772  1772
1794              do rwptr [ j ] := succ ( rwptr [ j ] )                401  1772     S   401  1772
1795            end
1796          ; rw [ i ] . symbol := fsymbol                            402  1772   406  1770
1797          ; rwct := succ ( rwct )                                   408     S   408
1798          end
1799        end { INITRW }
1300
1801    ; procedure initdumlrec ( var lrp : linerecptr )                  *     *   295
1802
1803      ; begin
1804          with lrp ^                                               1801
1805          do begin
1806            dlink := nil                                            298
1807          ; rlink := lrp                                            298  1801
1808          ; newline := false                                        303     S
1809        {% IF SINGLEPRECISION}
1810          ; tokno := 0                                              311
1811        {% END SINGLEPRECISION}
1812        {% IF MULTPRECISION}
1813          ; multzero ( tokno )                                      551   311
1814        {% END MULTPRECISION}
1815          ; symbol := tdummy                                        406     *
1816          ; startpos := leftmargin                                  301   267
1817          ; length := 0                                             302
1818        {% IF SINGLEPRECISION}
1819          ; aggsize := 0                                            299
1820        {% END SINGLEPRECISION}
1821        {% IF MULTPRECISION}
1822          ; multzero ( aggsize )                                    551   299
1823        {% END MULTPRECISION}
1824          end
1825        end { INITDUMLREC }
1826
1827    ; procedure initsets                                              *
1828
1329      ; begin
1830          letters                                                   409
1831          := [ 'A' , 'B' , 'C' , 'D' , 'E' , 'F' , 'G' , 'H' , 'I'
1832             , 'J' , 'K' , 'L' , 'M' , 'N' , 'O' , 'P' , 'Q' , 'R'
1833             , 'S' , 'T' , 'U' , 'V' , 'W' , 'X' , 'Y' , 'Z' , 'a'
1834             , 'b' , 'c' , 'd' , 'e' , 'f' , 'g' , 'h' , 'i' , 'j'
1835             , 'k' , 'l' , 'm' , 'n' , 'o' , 'p' , 'q' , 'r' , 's'
1836             , 't' , 'u' , 'v' , 'w' , 'x' , 'y' , 'z' ]
1837        ; octaldigits                                               409
1838          := [ '0' , '1' , '2' , '3' , '4' , '5' , '6' , '7' ]
1839        ; digits := octaldigits + [ '8' , '9' ]                     409   409
1840        ; hexdigits                                                 409
1841          := digits                                                 409
1842             + [ 'A' , 'B' , 'C' , 'D' , 'E' , 'F' , 'a' , 'b' , 'c'
1843               , 'd' , 'e' , 'f' ]
1844        ; lexchars                                                  409
1845          := letters                                                409
1846             + digits                                               409
1847             + [ '''' , '"' , '$' , '+' , '-' , '>' , '<' , '(' , ':'
```

```
1848                , '.' , ')' , '=' , '/' , '*' , '[' , ']' , ',' , '@'
1849                , ';' , '%' , ' ' , '(' , ')' , '_' , '!'
1850                {% IF ASCII}
1851                , '^' {% END ASCII} ]
1852          ; throwaways := [ chr ( ht ) , chr ( ff ) ]                    410     S   280     S   281
1853          end { INITSETS }
1854
1855       ; procedure initrwtable                                            *
1856
1857          ; begin
1858              rwct := 0                                                  408
1859            ; for i := 1 to 13 do rwptr [ i ] := 1                      1772   401  1772
1860            ; initrw ( tactivate , 'ACTIVATE     ' )                    1770     *
1861            ; initrw ( tafter , 'AFTER       ' )                        1770     *
1862            ; initrw ( tand , 'AND         ' )                          1770     *
1863            ; initrw ( tarray , 'ARRAY       ' )                        1770     *
1864            ; initrw ( tat , 'AT          ' )                           1770     *
1865            ; initrw ( tbefore , 'BEFORE      ' )                       1770     *
1866            ; initrw ( tbegin , 'BEGIN       ' )                        1770     *
1867            ; initrw ( tboolean , 'BOOLEAN     ' )                      1770     *
1868            ; initrw ( tcharacter , 'CHARACTER   ' )                    1770     *
1869            ; initrw ( tclass , 'CLASS       ' )                        1770     *
1870            ; initrw ( tcomment , 'COMMENT     ' )                      1770     *
1871            ; initrw ( tdelay , 'DELAY       ' )                        1770     *
1872            ; initrw ( tdo , 'DO          ' )                           1770     *
1873            ; initrw ( telse , 'ELSE        ' )                         1770     *
1874            ; initrw ( tend , 'END         ' )                          1770     *
1875            ; initrw ( teqv , 'EQV         ' )                          1770     *
1876            ; initrw ( texternal , 'EXTERNAL    ' )                     1770     *
1877            ; initrw ( tfor , 'FOR         ' )                          1770     *
1878            ; initrw ( tgo,   'GO          ' )                          1770     *
1879            ; initrw ( tgoto , 'GOTO        ' )                         1770     *
1880            ; initrw ( thidden , 'HIDDEN      ' )                       1770     *
1881            ; initrw ( tif , 'IF          ' )                           1770     *
1882            ; initrw ( timp , 'IMP         ' )                          1770     *
1883            ; initrw ( tin , 'IN          ' )                           1770     *
1884            ; initrw ( tinner , 'INNER       ' )                        1770     *
1885            ; initrw ( tinspect , 'INSPECT     ' )                      1770     *
1886            ; initrw ( tinteger , 'INTEGER     ' )                      1770     *
1887            ; initrw ( tis , 'IS          ' )                           1770     *
1888            ; initrw ( tlabel , 'LABEL       ' )                        1770     *
1889            ; initrw ( tlong , 'LONG        ' )                         1770     *
1890            ; initrw ( tname , 'NAME        ' )                         1770     *
1891            ; initrw ( tnew , 'NEW         ' )                          1770     *
1892            ; initrw ( tnone , 'NONE        ' )                         1770     *
1893            ; initrw ( tnot , 'NOT         ' )                          1770     *
1894            ; initrw ( tnotext , 'NOTEXT      ' )                       1770     *
1895            ; initrw ( tor , 'OR          ' )                           1770     *
1896            ; initrw ( totherwise , 'OTHERWISE   ' )                    1770     *
1897            ; initrw ( tprior , 'PRIOR       ' )                        1770     *
1898            ; initrw ( tprocedure , 'PROCEDURE   ' )                    1770     *
1899            ; initrw ( tprotected , 'PROTECTED   ' )                    1770     *
1900            ; initrw ( tqua , 'QUA         ' )                          1770     *
1901            ; initrw ( treactivate , 'REACTIVATE  ' )                   1770     *
1902            ; initrw ( treal , 'REAL        ' )                         1770     *
1903            ; initrw ( tref , 'REF         ' )                          1770     *
```

```
1904          ; initrw ( tshort , 'SHORT       ' )                          1770     *
1905          ; initrw ( tstep , 'STEP        ' )                           1770     *
1906          ; initrw ( tswitch , 'SWITCH      ' )                         1770     *
1907          ; initrw ( ttext , 'TEXT        ' )                           1770     *
1908          ; initrw ( tthen , 'THEN        ' )                           1770     *
1909          ; initrw ( tthis , 'THIS        ' )                           1770     *
1910          ; initrw ( tto , 'TO          ' )                             1770     *
1911          ; initrw ( tunknown , 'UNKNOWN     ' )                        1770     *
1912          ; initrw ( tuntil , 'UNTIL       ' )                         1770     *
1913          ; initrw ( tvalue , 'VALUE       ' )                         1770     *
1914          ; initrw ( tvirtual , 'VIRTUAL     ' )                       1770     *
1915          ; initrw ( twhen , 'WHEN        ' )                          1770     *
1916          ; initrw ( twhile , 'WHILE       ' )                         1770     *
1917          ; initrw ( txor , 'XOR         ' )                           1770     *
1918          ; initrw ( teq, 'EQ          ' )                             1770     *
1919          ; initrw ( tge, 'GE          ' )                             1770     *
1920          ; initrw ( tgt, 'GT          ' )                             1770     *
1921          ; initrw ( tlt, 'LT          ' )                             1770     *
1922          ; initrw ( tle, 'LE          ' )                             1770     *
1923          ; initrw ( tne, 'NE          ' )                             1770     *
1924          end { INITRWTABLE }
1925
1926        ; procedure initinp                                              *
1927
1928          ; var lstring : string12                                      939   320
1929            ; i , j : integer                                          1772  1772    27
1930
1931          ; begin
1932            {% IF PASCAL32}
1933              open ( initfile , 3 , opforin )                           456   417   326
1934            {% END PASCAL32}
1935            {% IF PASCAL}                        reset ( initfile :     S   417
1936            {% END PASCAL}
1937            ; get ( initfile )                                          422   417
1938            ; while not eof ( initfile )                                487   417
1939              do begin
1940                lstring :=  '            '                             1928
1941              ; i := 1                                                  1929
1942              ; while not eof ( initfile )                               487   417
1943                      and ( eoln ( initfile )                            491   417
1944                            or not ( inputwindow ( initfile ) in letters )  539   417   409
1945                    )
1946                do get ( initfile ) { SCAN FOR LETTER }                 422   417
1947            { ASSERT EOF OR (NOT EOLN AND ( INPUTWINDOW ( INITFILE ) IN LETTERS)) }
1948              ; if not eof ( initfile )                                 487   417
1949                then begin
1950                  while not eof ( initfile )                            487   417
1951                        and not eoln ( initfile )                       491   417
1952                        and ( inputwindow ( initfile ) in letters )     539   417   409
1953                        and ( i <= 12 )                                1929
1954                  do begin
1955                    lstring [ i ] := inputwindow ( initfile )          1928  1929   539   417
1956                  ; i := succ ( i )                                    1929     S  1929
1957                  ; get ( initfile )                                    422   417
1958                  end
1959              ; while not eof ( initfile )                              487   417
```

```
1960                    and not eoln ( initfile )                          491   417
1961                    and ( inputwindow ( initfile ) in letters )        539   417   409
1962              do get ( initfile )                                      422   417
1963            ( ASSERT: 1 <= I-1 = NO OF LETTERS <= 12 )
1964            ; for j := rwptr [ i - 1 ] to rwptr [ i ] - 1            1929   401  1929   401  1929
1965              do with rw [ j ]                                         402  1929
1966                do if string = lstring                                1009  1928
1967                  then symbol := tident                               406     *
1968            end
1969          end
1970        {% IF PASCAL32}
1971        ; close ( initfile )                                          521   417
1972        {% END PASCAL32}
1973        end
1974
1975 {SL Interpreter, written in Pascal.}
1976 {Jan 5, 1983 }
1977
1978    ; procedure SyntaxError ( sourceLineNo , SSLLineNo : integer )       *   726     *    27
1979
1980      ; begin
1981          if displaySyntaxErrors                                       345
1982          then begin
1983            DisplayString ( 'SyntaxErr, srcLine$' )                    712
1984          ; DisplayInteger ( sourceLineNo )                            693  1978
1985          ; DisplayString ( 'S/SLLine$' )                              712
1986          ; DisplayInteger ( SSLLineNo )                               693  1978
1987          ; DisplayString ( 'token$' )                                 712
1988          ; DisplayInteger ( gcursy )                                  693   389
1989          ; DisplayInteger ( gcurp ^ . tokno )                         693   385   311
1990          ; display ( nl )                                             142    33
1991          end
1992        end
1993
1994    ; procedure Check ( SSLLineNo : integer ; keys : setofsymbol )       *  1978    27   350   224
1995
1996            {This procedure checks if the incoming token is in the
c 1997            current recovery token set and emits an error message if not }
1998
1999      ; begin
2000          if not ( gcursy in keys )                                    389  1994
2001          then begin
2002            SyntaxError ( sourceLineNo , SSLLineNo )                  1978  1978  1994
2003          ; skip ( keys , displaySyntaxErrors )                       1691  1994   345
2004          end
2005        end {Check}
2006
2007    ; procedure ConvertToKeySet                                          *
2008        ( address : integer ; var result : setofsymbol )                 S    27   587   224
2009
2010      ; var lrec                                                         4
2011          : record
2012              case boolean                                             S
2013              of false : ( ary : SetArray )                            S     *   222
2014              ; true : ( pset : setofsymbol )                          S     *   224
2015            end
```

```
  2016         ; i : 1 .. setSize                                          1929   220
  2017
  2018       ; begin
  2019           for i := 0 to setSize                                     2016   220
  2020           do lrec . ary [ i ] := table [ address + i ]               2010  2013  2016     *  2008  2016
  2021         ; result := lrec . pset                                     2008  2010  2014
  2022         end {onvertToKeySet}
  2023
  2024     ; procedure HandleSemanticChoice ( choicetag : integer )          *     *    27
  2025
  2026             {This procedure performs
c 2027             choices.  It sequentially tests each alternative
c 2028             value against the tag value, and when a match is
c 2029             found, performs a branch to the corresponding
c 2030             alternative path.  If none of the alternative
c 2031             values matches the tag value, table interpretation
c 2032             proceeds to the operation immediately following
c 2033             the list of alternatives (normally the otherwise
c 2034             path).  }
  2035
  2036       ; var numberOfChoices : integer                                 *    27
  2037         ; choiceTableAddress : integer                                *    27
  2038
  2039       ; begin
  2040           choiceTableAddress := table [ tablePointer ]              2037    *   346
  2041         ; numberOfChoices := table [ choiceTableAddress ]           2036    *  2037
  2042         ; tablePointer := choiceTableAddress + 1                     346  2037
  2043         ; repeat if table [ tablePointer ] = choicetag                *    346  2024
  2044             then begin
  2045               tablePointer := table [ tablePointer + 1 ]             346    *   346
  2046             ; numberOfChoices := 0                                  2036
  2047             end
  2048             else begin
  2049               tablePointer := tablePointer + 2                       346   346
  2050             ; numberOfChoices := numberOfChoices - 1                2036  2036
  2051             end
  2052           until numberOfChoices = 0                                 2036
  2053         ;
  2054         end {andleSemanticChoice}
  2055
  2056     ; procedure HandleInputChoice                                     *
  2057         ( choicetag : integer ; var choiceTagMatched : Boolean )     2024    27     *     S
  2058
  2059             {This procedure performs input
c 2060             choices.  It sequentially tests each alternative
c 2061             value against the tag value, and when a match is
c 2062             found, performs a branch to the corresponding
c 2063             alternative path.  If none of the alternative
c 2064             values matches the tag value, table interpretation
c 2065             proceeds to the operation immediately following
c 2066             the list of alternatives (normally the otherwise
c 2067             path).  The flag choicetagMatched is set to true
c 2068             if a match is found and false otherwise.    }
  2069
  2070       ; var numberOfChoices : integer                               2036    27
  2071         ; choiceTableAddress : integer                              2037    27
```

```
  2072        ; SSLLineNo : integer                                          1994    27
  2073        ; keyAddress : integer                                            *    27
  2074        ; keys : setofsymbol                                           1994   224
  2075
  2076      ; begin
  2077          choiceTableAddress := table [ tablePointer ]                 2071     *   346
  2078        ; SSLLineNo := table [ tablePointer + 1 ]                      2072     *   346
  2079        ; keyAddress := table [ tablePointer + 2 ]                     2073     *   346
  2080        ; numberOfChoices := table [ choiceTableAddress ]              2070     *  2071
  2081        ; tablePointer := choiceTableAddress + 1                        346  2071
  2082        ; choicetagMatched := false                                     352  2013
  2083        ; ConvertToKeySet ( keyAddress , keys )                        2007  2073  2074
  2084        ; Check                                                        1994
  2085            ( SSLLineNo , keys + callStack [ callStackTop ] . sett )   2072  2074   366   365   228
  2086        ; repeat if table [ tablePointer ] = choicetag                    *   346  2057
  2087            then begin
  2088              choicetagMatched := true                                  352  2014
  2089            ; tablePointer := table [ tablePointer + 1 ]                346     *   346
  2090            ; numberOfChoices := 0                                     2070
  2091            end
  2092            else begin
  2093              tablePointer := tablePointer + 2                          346   346
  2094            ; numberOfChoices := numberOfChoices - 1                   2070  2070
  2095            end
  2096          until numberOfChoices = 0                                    2070
  2097        ;
  2098        end {andleInputChoice}
  2099
  2100        {Execution Tracing }
  2101
  2102    ; procedure Trace                                                     *
  2103
  2104            {mports (operation, table, tablePointer, var IO);}
  2105
  2106      ; begin
  2107              {O.PutString ('table index ');
c 2108               IO.PutInt (tablePointer-1);
c 2109               IO.PutString (';  Operation ');
c 2110               IO.PutInt (operation);
c 2111               IO.PutString (';  Argument ');
c 2112               IO.PutInt (table(tablePointer));
c 2113               IO.PutLine;}
  2114        end {race}
  2115
  2116        {Semantic Choice Failure }
  2117
  2118    ; procedure HandleSemanticChoiceFailure                               *
  2119
  2120            {mports (sourceLineNo, Trace, var IO, Abort);}
  2121
  2122      ; begin
  2123              {IO.PutString ('Semantic choice failed;  Line ');
c 2124               IO.PutInt (sourceLineNo);
c 2125               IO.PutInt ( table [ tablePointer ] );
c 2126               IO.PutLine; }
  2127        end {andleSemanticChoiceFailure}
```

```
2128
2129      ; procedure HandleMatchFailure                                          *
2130
2131              {This procedure handles syntax errors in the input
c 2132             to the S/SL program.
c 2133
c 2134             Syntax error recovery:
c 2135             When a mismatch occurs between the the input token
c 2136             and the operand of an input operation,
c 2137        the following recovery is employed.
c 2138
c 2139             The keyAddress is converted to keySet, and keys is
c 2140             created by the union of keySet and the set in the top
c 2141             callStack entry. An error message is emitted.  Incoming
c 2142             tokens are skipped until one is found belonging in keys. }
2143
2144        ; var keyAddress : integer                                    2073    27
2145          ; SSLLineNo : integer  {S/SL sourceLineNo }                 2072    27
2146          ; keys : setofsymbol                                        2074   224
2147          ; semicolonExpected : boolean                                  *  2012
2148
2149        ; begin
2150            SSLLineNo := table [ tablePointer + 1 ]                   2145     *   346
2151          ; keyAddress := table [ tablePointer + 2 ]                  2144     *   346
2152          ; semicolonExpected := table [ tablePointer ] = tSemicolon  2147     *   346     *
2153          ; ConvertToKeySet ( keyAddress , keys )                     2007  2144  2146
2154          ; if displaySyntaxErrors and not semicolonExpected           345  2147
2155            then SyntaxError ( sourceLineNo , SSLLineNo )             1978  1978  2145
2156          ; skip                                                      1691
2157              ( keys + callStack [ callStackTop ] . sett              2146   366   365   228
2158              , displaysyntaxErrors and not semicolonExpected )          *  2147
2159          end {andleMatchFailure}
2160
2161          {Main Walker Program }
2162
2163      ; procedure Interpreter                                             *
2164
2165                  {Walk the Syntax/Semantic table }
2166
2167        ; var keys : setofsymbol                                    2146   224
2168
2169        ; begin
2170            processing := true                                       342  2014
2171          ; tracing := false                                         343  2013
2172          ; semanticizing := true                                    344  2014
2173          ; tablePointer := 0                                        346
2174          ; callStackTop := 1                                        365
2175          ; callStack [ callStackTop ] . sett := [ tEndOfFile ]      366   365   228     *
2176          ; gError := 0                                              375
2177          ; while processing                                         342
2178            do begin
2179              operation := table [ tablePointer ]                    347     *   346
2180            ; tablePointer := tablePointer + 1                       346   346
2181                      {Trace Execution }
2182            ; if tracing                                             343
2183              then Trace                                            2102
```

```
2184          ; case operation                                          347
2185            of oRecovCall                                           198
2186              : begin
2187                if callStackTop < callStackSize                     365   233
2188                then begin
2189                  ConvertToKaySet                                  2007
2190                    ( table [ tablePointer + 1 ] , keys )             *   346  2167
2191                ; callStackTop := callStackTop + 1                  365   365
2192                ; callStack [ callStackTop ] . sett                 366   365   228
2193                  := keys + callStack [ callStackTop - 1 ] . sett  2167   366   365   228
2194                ; callStack [ callStackTop ] . returnAddress        366   365   230
2195                  := tablePointer + 2                               346
2196                ; tablePointer := table [ tablePointer ]            346     *   346
2197                end
2198                else begin
2199                  EmitError ( eCallStackOverflow , sourceLineNo )   725   244  1978
2200                ; processing := false                               342  2013
2201                end
2202            end {RecovCall}
2203          ; oReturn                                                 190
2204            : begin
2205              if callStackTop = 1                                   365
2206              then
2207                              {Return from main S/SL procedure }
2208                processing := false                                 342  2013
2209              else begin
2210                tablePointer                                        346
2211                := callStack [ callStackTop ] . returnAddress       366   365   230
2212              ; callStackTop := callStackTop - 1                    365   365
2213              end
2214            end {Return}
2215          ; oRepeat , oMerge                                        191   192
2216            : begin
2217              tablePointer := table [ tablePointer ]                346     *   346
2218            end {Repeat , oMerge }
2219          ; oRecovInput                                             199
2220            : begin
2221              if setupState <> unmatched                            388   388
2222              then EmitError ( eNotSetup , sourceLineNo )           725   251  1978
2223            ; if table [ tablePointer ] = gcursy                      *   346   389
2224              then setupState := matched                            388   398
2225              else begin {Syntax Error in Input }
2226                HandleMatchFailure                                 2129
2227              ; setupState := matchFailed                           388   388
2228              end
2229            ; tablePointer := tablePointer + 3                      346   346
2230            end {RecovInput}
2231          ; oRecovInputChoice                                       200
2232            : begin
2233              if setupState <> unmatched                            389   388
2234              then EmitError ( eNotSetup , sourceLineNo )           725   251  1978
2235            ; HandleInputchoice ( gcursy , choicetagMatched )         *   389   352
2236            ; if choicetagMatched                                   352
2237              then setupState := matched                            398   388
2238            end {RecovInputChoice}
2239          ; oRecovChoiceFailure                                     201
```

```
2240              : begin
2241                SyntaxError ( sourceLineNo , table [ tablePointer ] )     1978  1978     *   346
2242              ; tablePointer := tablePointer + 1                           346   346
2243              ; setupState := matchFailed                                  388   388
2244              end {RecovChoiceFailure}
2245            {The Following are Pass dependent
c 2246            Data Structure Semantic Operations  }
2247            ; oSetupGcurPosNc                                               *
2248              : begin
2249                setup ( gcurposnc , false )                               1518   390  2013
2250              end {oSetupGcurPosNc}
2251            ; oSetupPos0                                                    *
2252              : begin
2253                setup ( PosStack [ PosStackTop ] , false )                1518   394   396  2013
2254              end {oSetupPos0}
2255            ; oSetupPos1                                                    *
2256              : begin
2257                setup ( PosStack [ PosStackTop - 1 ] , false )            1518   394   396  2013
2258              end {oSetupPos1}
2259            ; oSetupIndentPos0                                              *
2260              : begin
2261                setup ( indent ( PosStack [ PosStackTop ] ) , false )     1518  1759   394   396  2013
2262              end {oSetupIndentPos0}
2263            ; oCcompress0                                                   *
2264              : begin
2265                ccompress ( Markstack [ MarkStackTop ] )                  1562     *   397
2266              end {oCcompress0}
2267            ; oCcompress1                                                   *
2268              : begin
2269                ccompress ( Markstack [ MarkStackTop - 1 ] )              1562     *   397
2270              end {oCcompress1}
2271            ; oCcompress2                                                   *
2272              : begin
2273                ccompress ( Markstack [ MarkStackTop - 2 ] )              1562     *   397
2274              end {oCcompress2}
2275            ; oPushConstPos                                                 *
2276              : begin
2277                PushConstPos ( table [ tablePointer ] )                   1444     *   346
2278              ; tablePointer := tablePointer + 1                           346   346
2279              end {oPushConstPos}
2280            ; oPushGcurPosNc                                                *
2281              : begin
2282                PushPos ( gcurposnc )                                     1458   390
2283              end {oPushGcurPosNc}
2284            ; oPushIndentPos0                                               *
2285              : begin
2286                PushPos ( indent ( PosStack [ PosStackTop ] ) )           1458  1759   394   396
2287              end {oPushIndentPos0}
2288            ; oPushIndentPos1                                               *
2289              : begin
2290                PushPos ( indent ( PosStack [ PosStackTop - 1 ] ) )       1458  1759   394   396
2291              end {oPushIndentPos1}
2292            ; oPushCopyPos0                                                 *
2293              : begin
2294                PushPos ( PosStack [ PosStackTop ] )                      1458   394   396
2295              end {oPushCopyPos0}
```

```
2296              ; oPushCopyPos1                                          *
2297                : begin
2298                  PushPos ( PosStack [ PosStackTop - 1 ] )          1458   394   396
2299                end {oPushCopyPos1}
2300              ; oPushCopyMark0                                         *
2301                : begin
2302                  PushMark ( MarkStack [ MarkStackTop ] )           1472   395   397
2303                end {oPushCopyMark0}
2304              ; oPushCopyMark1                                         *
2305                : begin
2306                  PushMark ( MarkStack [ MarkStackTop - 1 ] )       1472   395   397
2307                end {oPushCopyMark1}
2308              ; oPushNextTokenMark                                     *
2309                : begin
2310                  mark ( Lmark , gnextp )                            783   351   385
2311                ; PushMark ( Lmark )                               1472   351
2312                end {oPushNextTokenMark}
2313              ; oPushCurTokenMark                                      *
2314                : begin
2315                  mark ( Lmark , gcurp )                             783   351   385
2316                ; PushMark ( Lmark )                               1472   351
2317                end {oPushCurTokenMark}
2318              ; oPopPosAndMark                                         *
2319                : begin
2320                  PopPos                                           1486
2321                ; PopMark                                          1497
2322                end {oPopPosAndMark}
2323              ; oPopPos : begin PopPos end {oPopPos}                   *  1486
2324              ; oMoveCmnts                                             *
2325                : begin
2326                  if setupState = matched                           388   388
2327                  then movecmnts                                   1634
2328                end {oMoveCmnts}
2329              ; oPopMark : begin PopMark end {oPopMark}                *  1497
2330              ; oSwapMarks : begin SwapMarks end {oSwapMarks}          *  1508
2331              ; oBlankLine : begin blankline end {oBlankLine}          *  1267
2332              ; oChooseCurInputToken                                   *
2333                : HandleSemanticChoice ( gcursy )                  2024   389
2334                {oChooseCurInputToken}
2335              ; oChooseNextInputToken                                  *
2336                : HandleSemanticChoice ( gnextp ^ . symbol )       2024   385   406
2337                {oChooseNextInputToken}
2338              ; oMatchCmnt                                             *
2339                : setupState := matched {oMatchCmnt}                388   388
2340              end {case}
2341            end {loop}
2342          end {Interpreter}
2343
2344      ; begin { MAIN BLOCK }
2345      {% IF PASCAL32}
2346          displaySyntaxErrors := false                               345  2013
2347        ; i := 1                                                    2016
2348        ; while ( i <= linelength ) and ( param [ i ] <> nl )       2016    42   171  2016    33
2349          do begin
2350            if param [ i ] in [ 'B' , 'b' ]                          171  2016
2351            then breakpnt                                            419
```

```
2352          else if param [ i ] in [ 'T' , 't' ]                          171  2016
2353            then display3yntaxErrors := true                            345  2014
2354        ; i := succ ( i )                                              2016     S  2016
2355        end
2356      ; open ( input , 1 , opforin )                                    456   417   326
2357      ; open ( output , 2 , opforout )                                  456   417   326
2358    {% END PASCAL32}
2359    {% IF PASCAL}
2360      ; reset ( input )                                                   S   417
2361      ; rewrite ( output )                                                S   417
2362    {% END PASCAL}
2363      ; initsets                                                       1827
2364      ; initrwtable                                                    1855
2365    {; initinp   }
2366      ; pos := 1              { INITIALIZATIONS FOR INSYMBOL }          378
2367      ; getlrec ( gnextp )                                              759   385
2368      ; initdumlrec ( gnextp )                                         1801   385
2369    {% IF SINGLEPRECISION}
2370      ; gnextp ^ . tokno := 2                                           385   311
2371    {% END SINGLEPRECISION}
2372    {% IF MULTPRECISION}
2373      ; multzero ( gnextp ^ . tokno )                                   551   385   311
2374      ; multplint ( gnextp ^ . tokno , 2 , gnextp ^ . tokno )           557   385   311   385   311
2375    {% END MULTPRECISION}
2376      ; getlrec ( gcurp )                                               759   385
2377      ; initdumlrec ( gcurp )                                          1801   385
2378    {% IF SINGLEPRECISION}
2379      ; gcurp ^ . tokno := 1                                            385   311
2380    {% END SINGLEPRECISION}
2381    {% IF MULTPRECISION}
2382      ; multzero ( gcurp ^ . tokno )                                    551   385   311
2383      ; multplint ( gcurp ^ . tokno , 1 , gnextp ^ . tokno )            557   385   311   385   311
2384    {% END MULTPRECISION}
2385      ; gfirstp := gcurp                                                385   385
2386      ; gcurp ^ . dlink := gnextp                                       385   298   385
2387      ; gtrailp := nil                                                  385
2388      ; gcompp := gcurp                                                 385   385
2389      ; gcomptrailp := nil                                              385
2390      ; gfreep := nil                                                   386
2391      ; outpos := leftmargin                                            390   267
2392      ; gcurpos := leftmargin                                           390   267
2393      ; gcurposnc := leftmargin                                         390   267
2394      ; gcursy := tUnknown                                              389     *
2395      ; rbracksw := false                                               380  2013
2396      ; elipsissw := false                                              380  2013
2397      ; innewline := true                                               379  2014
2398      ; cmntstate := nocomment                                          381   381
2399    {% IF SINGLEPRECISION}
2400      ; gaggsize := 0                                                   393
2401      ; gtokno := 2                                                     392
2402      ; nltokno := 2                                                    392
2403      ; outtokno := 0                                                   392
2404    {% END SINGLEPRECISION}
2405    {% IF MULTPRECISION}
2406      ; multzero ( gaggsize )                                           551   393
2407      ; multzero ( gtokno )                                             551   392
```

```
2408        ; multplint ( gtokno , 2 , gtokno )                                  557   392   392
2409        ; nltokno := gtokno                                                  392   392
2410        ; multzero ( outtokno )                                              551   392
2411        ; multzero ( linespace )                                             551   399
2412        ; multplint                                                          557
2413            ( linespace , rightmargin - leftmargin + 1 , linespace )         399   268   267   399
2414        ; multzero ( multrightmargin )                                       551   399
2415        ; multplint ( multrightmargin , rightmargin , multrightmargin )      557   399   268   399
2416      {% END MULTPRECISION}
2417        ; sourceLineNo := 0                                                 1978
2418        ; lineoutct := 0                                                     412
2419        ; linerecct := 0                                                     412
2420        ; tokenct := 0                                                       412
2421        ; skipct := 0                                                        412
2422        ; gpos := leftmargin                                                 390   267
2423        ; mark ( gmark , gcurp )                                             783   391   385
2424        ; insymbol ( false )                                                1003  2013
2425        ; insymbol ( false )                                                1003  2013
2426        ; setupState := unmatched                                            388   388
2427        ; PosStackTop := 0                                                   396
2428        ; MarkStackTop := 0                                                  397
2429        ; Interpreter                                                       2163
2430        ; unload                                                            1367
2431        ; writeln ( output )                                                 517   417
2432        ; writeln ( output )                                                 517   417
2433        ; lineoutct := lineoutct + 2                                         412   412
2434      {% IF PASCAL32}
2435        ; close ( input )                                                    521   417
2436        ; close ( output )                                                   521   417
2437        ; open ( summary , 0 , opforout )                                    456   417   326
2438      {% END PASCAL32}
2439      {% IF PASCAL}
2440        ; rewrite ( summary )                                                  S   417
2441      {% END PASCAL}
2442        ; reportstat ( sourceLineNo , 'INPUT LINES ' )                       680  1978
2443        ; reportstat ( lineoutct , 'OUTPUT LINES' )                          680   412
2444        ; reportstat ( tokenct , 'TOKENS       ' )                           680   412
2445        ; reportstat ( skipct , 'SKIPPED      ' )                            680   412
2446        ; reportstat ( linerecct , 'HEAP RECORDS' )                          680   412
2447      {% IF PASCAL32}
2448        ; close ( summary )                                                  521   417
2449      {% END PASCAL32}
2450        end ( MAIN BLOCK )
2451 .
```

cross reference * is def = is asg

```
 trefEqual              1151
 trefNotEqual           1159
 trightParen            1170
 true                    447   449   599   632   835   840   883   954   960  1025  1051  1255  1714  1746  2014* 2088
                        2170  2172  2353  2397
 tSemicolon             2152
 tsemicolon             1172
 tshort                 1904
 tsimplestring          1067
 tstep                  1905
 tswitch                1906
 tSyntaxError            263*
 ttext                  1907
 tthen                  1908
 tthis                  1909
 ttimes                 1136
 tto                    1910
 tUnknown               2394
 tunknown               1193  1911
 tuntil                 1912
 tvalue                 1913
 tvirtual               1914
 twhen                  1915
 twhile                 1916
 tXor                    265
 txor                   1917
 typedevice               63*

-U-
 unload                  780* 1367* 1630  1681  1686  1738  1743  2430
 unmatched               388* 1264  1524  1526  1556  2221  2233  2426
 upperCase               927*  932=  934=  949
 upspace                  68*

-V-
 varianterror            115*

-W-
 work                    560*  570=  571=  571   572=  572   573   589*  602=  603=  603   604=  604   605=  605   606
                         615=  616=  616   617=  617   618=  618   619   642*  645   646   651*  654   655   655   656
 write                   495*  519   530   687   688   689  1414  1423
 writeeof                 68*
 writeln                 517*  690  1404  2431  2432

end xref  595 identifiers  3203 total references
2409 collisions.
```

Appendix 2

Syntax/Semantic Language code

```
1
2 %       SIMULA prettyprinter parser in SSL
3 %
4 %                    by
5 %
6 %              Jung-Juin Chen
7 %
8 %               November 1985
9
10
11
12
13
14 inputs :                                                          *
15
16
17
18          tactivate       'activate'                              *     *
19          tafter          'after'                                 *     *
20          tand            'and'                                   *     *
21          tarray          'array'                                 *     *
22          tat             'at'                                    *     *
23          tbasisOfTen     'basisOfTen'                            *
24          tbecomes        'becomes'                               *     *     *
25          tbefore         'before'                                *     *
26          tbegin          'begin'                                 *     *
27          tblankline                                              *
28          tboolean        'boolean'                               *     *
29          tcharacter      'character'                             *     *
30          tcharsequence   'charsequence'                          *     *
31          tclass          'class'                                 *     *
32          tcolon          'colon'                                 *     *
33          tcomma          'comma'                                 *     *
34          tcomment        'comment'                               *     *
35          tcomment1                                               *
36          tcomment2                                               *
37          tcomment3                                               *
38          tcomment4                                               *
39          tdelay          'delay'                                 *     *
40          tdenotes        'denotes'                               *     *
41          tdivide         'divide'                                *     *
42          tdo             'do'                                    *     *
43          tdot            'dot'                                   *     *
44          tdummy                                                  *
45          telse           'else'                                  *     *
46          tend            'end'                                   *     *
47          teq             'eq'                                    *     *
48          tendofFile      'endofFile'                             *
49          teqv            'eqv'                                   *     *     *
50          texpon          'expon'                                 *
51          texternal       'external'                              *     *     *
52          tfor            'for'                                   *     *
53          tge             'ge'                                    *
54          tgt             'gt'                                    *     *
55          tgo             'go'                                    *     *
```

```
56      tgoto           'goto'                                  *   *
57      thidden         'hidden'                                *   *   *
58      tident          'ident'                                 *   *
59      tif             'if'                                    *   *
60      timp            'imp'                                   *   *
61      tin             'in'                                    *   *
62      tinner          'inner'                                 *
63      tinspect        'inspect'                               *   *   *
64      tinteger        'integer'                               *   *
65      tis             'is'                                    *   *
66      tlabel          'label'                                 *   *
67      tle             'le'                                    *   *
68      tleftParen      'leftParen'                             *   *
69      tlong           'long'                                  *   *
70      tlt             'lt'                                    *   *
71      tminus          'minus'                                 *   *
72      tname           'name'                                  *   *
73      tne             'ne'                                    *   *
74      tnew            'new'                                   *   *
75      tnone           'none'                                  *
76      tnot            'not'                                   *   *   *
77      tnotext         'notext'                                *   *
78      tnumber         'number'                                *   *
79      tor             'or'                                    *   *
80      totherwise      'otherwise'                             *
31      tplus           'plus'                                  *   *
82      tprior          'prior'                                 *   *
83      tprocedure      'procedure'                             *   *   *
84      tprotected      'protected'                             *   *
85      tqua            'qua'                                   *   *
86      treactivate     'reactivate'                            *   *
87      treal           'real'                                  *   *
88      tref            'ref'                                   *   *
89      trefEqual       'refEqual'                              *   *
90      trefNotequal    'refNotequal'                           *   *
91      trightParen     'rightParen'                            *   *
92      tsemicolon      'semicolon'                             *   *
93      tshort          'short'                                 *   *
94      tsimpleString   'simpleString'                          *   *
95      tstep           'step'                                  *   *
96      tswitch         'switch'                                *   *
97      ttext           'text'                                  *   *
98      tthen           'then'                                  *
99      tthis           'this'                                  *   *   *
100     ttimes          'times'                                 *   *
101     tto             'to'                                    *
102     tunknown                                                *   *
103     tuntil          'until'                                 *   *
104     tvalue          'value'                                 *   *
105     tvirtual        'virtual'                               *   *
106     twhen           'when'                                  *   *
107     twhile          'while'                                 *   *
108     txor                                                    *
109     ;
110
111
```

```
112
113
114
115
116
117 SimpleOps :
118
119         oSetupGCurPosNc                                         *
120         oPushGCurPosNc                                          *
121         oPushCurTokenMark                                       *
122         oPushNextTokenMark                                      *
123         oPopPosAndMark                                          *
124         oPopPos                                                 *
125         oPopMark                                                *
126         oMoveCmnts                                              *
127         oBlankLine                                              *
128         oSwapMarks                                              *
129         oMatchCmnt                                              *
130
131         oPushIndentPos0                                         *
132         oPushIndentPos1                                         *
133         oPushCopyPos0                                           *
134         oPushCopyPos1                                           *
135         oPushCopyMark0                                          *
136         oPushCopyMark1                                          *
137         oSetupIndentPos0                                        *
138         oSetupPos0                                              *
139         oSetupPos1                                              *
140         oCcompress0                                             *
141         oCcompress1                                             *
142         oCcompress2                                             *
143         ;
144
145 ParmOps :
146
147         oPushConstPos                                           *
148         ;
149
150 ChoiceOps :
151
152         oChooseCurInputToken                                    *
153         oChooseNextInputToken                                   *
154         ;
155
156 _|_                                                             *     *
157
158
159
160
161
162
163
164
165
166
167
```

```
168 Programs :                                                          *
169
170         oPushConstPos(1)                                            147
171         oPushCurTokenMark                                           121
172         {[=
173            | 'endofFile' : >                                         49
174            | * :
175                 oPushCopyMark0                                      135
176                 oPushIndentPos0                                     131
177                 @SourceModule                                       *
178         ]}
179         oPopPosAndMark ;                                            123
180
181 % end of Programs
182
183
184
185
186 SourceModule :                                                      *
187
188         oPushCopyMark0                                              135
189         oPushCopyPos0                                               133
190         @ExternalDeclaration                                        *
191         [=
192            |'semicolon' :                                           92
193                 oPushCurTokenMark                                   121
194                 oMoveCmnts                                          126
195                 [oChooseNextInputToken                              153
196                    |'begin' :                                       26
197                         oBlankLine                                  127
198                    | * :
199                 ]
200                 oSetupPos1                                          139
201            | * :
202                 oPushCopyMark0                                      135
203                 oBlankLine                                          127
204         ]
205         oPushCopyMark0                                              135
206         oPushCopyPos0                                               133
207         @IdentList                                                  *
208         [oChooseCurInputToken                                       152
209            |'begin' :                                               26
210                 oPushCopyMark0                                      135
211                 oPushCopyPos0                                       133
212                 @UnconditionalStatement                             *
213            |'class' :                                               31
214                 oPushCopyMark0                                      135
215                 oPushCopyPos0                                       133
216                 @ClassDeclaration                                   *
217            | * :
218                 oPushCopyMark0                                      135
219                 oPushCopyPos0                                       133
220                 @Type                                               *
221                 oPushCopyMark0                                      135
222                 oPushCopyPos0                                       133
223                 @ProcedureDeclaration                               *
```

```
224         ]
225         oPopMark                                                125
226         oPopPosAndMark ;                                        123
227
228 % end of SourceModule
229
230
231
232
233 ExternalDeclaration :                                           *
234
235         {[=
236            |'external' :                                        51
237                 oPushCurTokenMark                               121
238                 oSetupPos0                                      138
239                 oCcompress1                                     141
240                 [=
241                    |'class' :                                   31
242                         oSetupGCurPosNc                         119
243                    | * :
244                         [=
245                            |'ident' :                           58
246                                 oSetupIndentPos0                137
247                                 oCcompress0                     140
248                            | * :
249                         ]
250                         oPushIndentPos0                         131
251                         oPushCurTokenMark                       121
252                         @Type                                   *
253                         oCcompress0                             140
254                         'procedure'
255                         oPopMark                                83    125
256                         oPushCurTokenMark                       121
257                         oSetupPos0                              138
258                 ]
259                 oPushIndentPos0                                 131
260                 oPushIndentPos0                                 131
261                 oPushCopyMark0                                  135
262                 oPushCurTokenMark                               121
263                 {
264                  'ident'
265                  oPushCurTokenMark                              58    121
266                  oSetupPos0                                     138
267                  [=
268                    |'eq' :                                      47
269                         oPushCurTokenMark                       121
270                         oSetupPos0                              138
271                         oPushCopyMark0                          135
272                         oPushCopyPos0                           133
273                         @String                                 *
274                         oCcompress0                             140
275                         oPopMark                                125
276                         oCcompress0                             140
277                         oPopMark                                125
278                    | * :
279                         oPopMark                                125
```

```
280                  ]
281                  oCcompress0                                    140
282                  oCcompress1                                    141
283                  oSwapMarks                                     128
284                  oPopMark                                       125
285                  [=
286                    |'comma' :                                    33
287                         oPushCurTokenMark                       121
288                         oMoveCmnts                              126
289                         oSetupPos1                              139
290                    | * :
291                         >
292                  ]
293                 }
294                 oPopPosAndMark                                  123
295                 oPopPos                                         124
296                 [=
297                    |'is' :                                       65
298                         oPushCurTokenMark                       121
299                         oSetupPos0                              138
300                         oPushCurTokenMark                       121
301                         oPushIndentPos0                         131
302                         @Type                                     *
303                         oCcompress0                             140
304                         oPushCopyMark0                          135
305                         oPushIndentPos0                         131
306                         @ProcedureDeclaration                     *
307                         oCcompress0                             140
308                         oPopMark                                125
309                         oCcompress0                             140
310                    | * :
311                 ]
312                 oPopMark                                        125
313            | * :
314                 >
315          ]
316         }
317         oPopPosAndMark ;                                        123
318
319 % end of ExternalDeclaration
320
321
322
323
324 UnconditionalStatement :                                          *
325
326         [=
327            |'bagin' :                                            26
328                 oPushCurTokenMark                               121
329                 oSetupPos0                                      138
330                 oCcompress1                                     141
331                 oPushCurTokenMark                               121
332                 oPushCopyPos0                                   133
333                 @Declaration                                      *
334                 {[=
335                    |'semicolon' :                                92
```

```
336                         oPushCurTokenMark                       121
337                         oMoveCmnts                              126
338                         oBlankLine                              127
339                         oSetupPos0                              138
340                         oPushCopyMark0                          135
341                         oPushCopyPos0                           133
342                         @Declaration                              *
343                         oCcompress0                             140
344                         oPopMark                                125
345                      | * :
346                           >
347                   ]
348                   }
349                   oPushCurTokenMark                             121
350                   oPushCopyPos0                                 133
351                   @StatementList                                  *
352                   oPopMark                                      125
353             |'goto' :                                            57
354                   oPushCurTokenMark                             121
355                   oSetupPos0                                    138
356                   oCcompress1                                   141
357                   oPushCopyMark0                                135
358                   oPushIndentPos0                               131
359                   @Expression                                     *
360                   oCcompress0                                   140
361                   oPopMark                                      125
362             |'go' :                                              56
363                   oPushCurTokenMark                             121
364                   oSetupPos0                                    138
365                   oCcompress1                                   141
366                   'to'
367                   oSetupGCurPosNc                               102   119
368                   oPushCopyMark0                                135
369                   oPushIndentPos0                               131
370                   @Expression                                     *
371                   oCcompress0                                   140
372                   oPopMark                                      125
373             |'activate','reactivate' :                           18    86
374                   oPushCurTokenMark                             121
375                   oSetupPos0                                    138
376                   oCcompress1                                   141
377                   oPushCurTokenMark                             121
378                   oPushIndentPos0                               131
379                   @Expression                                     *
380                   oCcompress0                                   140
381                   [=
382                      |'before', 'after', 'at', 'delay' :         25    19    22    39
383                            oPushCurTokenMark                     121
384                            oSetupPos0                            138
385                            oPushCurTokenMark                     121
386                            oPushIndentPos0                       131
387                            @Expression                             *
388                            oCcompress0                           140
389                            oPopMark                              125
390                            oCcompress0                           140
391                            [=
```

```
392                              |'prior' :                                  83
393                                   oSetupPos0                           139
394                                   oCcompress0                          140
395                              | * :
396                              ]
397                         | * :
398                    ]
399                    oPopMark                                             125
400               | * :
401                    oPushCurTokenMark                                    121
402                    oPushCurTokenMark                                    121
403                    oPushCopyPos0                                        133
404                    @Expression                                           *
405                    oCcompress1                                          141
406                    ([=
407                       |'becomes', 'denotes' :                            24    40
408                            oPushCurTokenMark                            121
409                            oSetupPos0                                   138
410                       | *  :
411                            >
412                     ]
413                     oPushCopyMark0                                      135
414                     oPushCopyPos0                                       133
415                     @Expression                                          *
416                     oCcompress0                                         140
417                     oCcompress1                                         141
418                     oSwapMarks                                          128
419                     oPopMark                                            125
420                    )
421                    oPopmark                                              *
422          ]
423          oPopPosAndMark ;                                              123
424
425 % end of UnconditionalStatement
426
427
428
429
430 ClassDeclaration :                                                      *
431
432          [=
433             |'ident' :                                                  59
434                oPushCurTokenMark                                       121
435                oSetupPos0                                              138
436                oCcompress1                                             141
437             | * :
438                oPushCopyMark0                                          135
439          ]
440          'class'
441          oPushCurTokenMark                                             31    121
442          oSetupPos0                                                    138
443          oCcompress1                                                   141
444          oSwapMarks                                                    128
445          oPopMark                                                      125
446          oPushCopyMark0                                                135
447          oPushCopyPos0                                                 133
```

```
448         @DeclarationPart                                          *
449         oCcompress0                                             140
450         'semicolon'
451         [oChooseNextInputToken                                   92    153
452            |'begin' :                                            26
453                oBlankLine                                       127
454            | * :
455         ]
456         oPushCurTokenMark                                       121
457         oMoveCmnts                                              126
458         oSetupPos0                                              138
459         [=
460            |'value' :                                           104
461                 oPushCurTokenMark                               121
462                 oSetupIndentPos0                                137
463                 oPushCurTokenMark                               121
464                 oPushIndentPos0                                 131
465                 @IdentList                                        *
466                 oCcompress0                                     140
467                 oPopMark                                        125
468                 oCcompress0                                     140
469                 oPopMark                                        125
470                 oCcompress0                                     140
471                 oPopMark                                        125
472                 'semicolon'                                      92
473                 oPushCurTokenMark                               121
474                 oMoveCmnts                                      126
475                 [oChooseNextInputToken                          153
476                    |'begin' :                                    26
477                         oBlankLine                              127
478                    | *:
479                 ]
480                 oSetupPos0                                      138
481            | * :
482                 oSwapMarks                                      128
483                 oPopMark                                        125
484         ]
485         {[oChooseCurInputToken                                  152
486            |'integer','real','boolean','character','text',       64    87    28    29    97
487             'short','long','ref' :                               93    69    98
488                  oPushCopyMark0                                 135
489                  oPushIndentPos0                                131
490                  @Type                                            *
491                  oCcompress0                                    140
492                  oPushCurTokenMark                              121
493                  oPushIndentPos0                                131
494                  [=
495                     |'array', 'procedure' :                      21    83
496                         oSetupPos0                              138
497                         oCcompress1                             141
498                     | * :
499                   ]
500                  oPushCopyMark0                                 135
501                  oPushIndentPos0                                131
502                  @IdentList                                       *
503                  oCcompress0                                    140
```

```
504                 oPopPosAndMark                                  123
505                 oCcompress0                                     140
506                 'semicolon'                                      92
507                 oPopMark                                        125
508                 oPushCurTokenMark                               121
509                 oMoveCmnts                                      126
510                 [oChooseNextInputToken                          153
511                    |'begin' :                                     26
512                         oBlankLine                              127
513                    | * :
514                 ]
515                 oSetupPos0                                      138
516             |'array','procedure','label','switch' :              21    83    66    96
517                 [=
518                    |'array','procedure','label','switch':       21    83    66    96
519                 ]
520                 oPushCurTokanMark                               121
521                 oPushIndentPos0                                 131
522                 oSetupPos0                                      138
523                 oCcompress1                                     141
524                 oPushCopyMark0                                  135
525                 oPushIndentPos0                                 131
526                 @IdentList                                       *
527                 oCcompress0                                     140
528                 oPopPosAndMark                                  123
529                 oCcompress0                                     140
530                 'semicolon'                                      92
531                 oPopMark                                        125
532                 oPushCurTokenMark                               121
533                 oMoveCmnts                                      126
534                 [oChooseNextInputToken                          153
535                    |'begin' :                                     26
536                         oBlankLine                              127
537                    | * :
538                 ]
539                 oSetupPos0                                      138
540             | * :
541                 >
542          ]
543         }
544         oPushCopyMark0                                          135
545         {
546          [=
547            |'hidden', 'protected' :                             57    84
548                 oSetupIndentPos0                                137
549                 oCcompress0                                     140
550                 [=
551                    |'hidden', 'protected' :                     57    84
552                         oSetupIndentPos0                        137
553                         oCcompress0                             140
554                    | * :
555                 ]
556                 oPushCopyMark0                                  135
557                 oPushIndentPos0                                 131
558                 oPushIndentPos0                                 131
559                 @IdentList                                       *
```

```
560                 oCcompress0                                   140
561                 oCcompress1                                   141
562                 oSwapMarks                                    128
563                 oPopPosAndMark                                123
564                 'semicolon'                                    92
565                 oPushCurTokenMark                             121
566                 oMoveCmnts                                    126
567                 [oChooseNextInputToken                        153
568                    |'begin' :                                  26
569                         oBlankLine                            127
570                    | * :
571                 ]
572                 oSetupPos0                                    138
573              | * :
574                 >
575           ]
576        }
577        oSwapMarks                                             128
578        oPopMark                                               125
579        [=
580           |'virtual' :                                        105
581                 oSetupPos0                                    138
582                 oCcompress0                                   140
583                 'colon'
584                 oPushCurTokenMark                              32   121
585                 oSetupPos0                                    138
586                 {[oChooseCurInputToken                        152
587                    |'integer','real','boolean','character','text',   64    87    28    29    97
588                     'short','long','ref' :                     93    69    88
589                         oPushCopyMark0                        135
590                         oPushIndentPos0                       131
591                         @Type                                  *
592                         oCcompress0                           140
593                         oPushCurTokenMark                     121
594                         oPushIndentPos0                       131
595                         [=
596                            |'array', 'procedure' :             21    83
597                                 oSetupPos0                    138
598                                 oCcompress1                   141
599                            | * :
600                         ]
601                         oPushCopyMark0                        135
602                         oPushIndentPos0                       131
603                         @IdentList                             *
604                         oCcompress0                           140
605                         oPopPosAndMark                        123
606                         oCcompress0                           140
607                         'semicolon'                            92
608                         oPopMark                              125
609                         oPushCurTokenMark                     121
610                         oMoveCmnts                            126
611                         [oChooseNextInputToken                153
612                            |'begin' :                          26
613                                 oBlankLine                    127
614                            | * :
615                         ]
```

```
616                     oSetupPos0                                      138
617                 |'array','procedure','label','switch' :             21    83    66    96
618                     [=
619                        |'array','procedure','label','switch':       21    83    66    96
620                     ]
621                     oPushCurTokenMark                               121
622                     oPushIndentPos0                                 131
623                     oSetupPos0                                      138
624                     oCcompress1                                     141
625                     oPushCopyMark0                                  135
626                     oPushIndantPos0                                 131
627                     @IdentList                                       *
628                     oCcompress0                                     140
629                     oPopPosAndMark                                  123
630                     oCcompress0                                     140
631                     'semicolon'                                      92
632                     oPopMark                                        125
633                     oPushCurTokenMark                               121
634                     oMoveCmnts                                      126
635                     [oChooseNextInputToken                          153
636                        |'begin' :                                    26
637                            oBlankLine                               127
638                        | * :
639                     ]
640                     oSetupPos0                                      138
641                 | * :
642                     >
643             ]
644             }
645             oSwapMarks                                              128
646             oPopmark                                                 *
647         | * :
648     ]
649     oPushIndentPos0                                                 131
650     [=
651        |'begin' :                                                    26
652             oSetupPos0                                              138
653             oCcompress0                                             140
654             oPushCurTokenMark                                       121
655             oPushCopyPos0                                           133
656             @Declaration                                             *
657             ([=
658                |'semicolon' :                                        92
659                     oPushCurTokenMark                               121
660                     oMoveCmnts                                      126
661                     oBlankLine                                      127
662                     oSetupPos0                                      138
663                     oPushCopyMark0                                  135
664                     oPushCopyPos0                                   133
665                     @Declaration                                     *
666                     oCcompress0                                     140
667                     oPopMark                                        125
668                 | * :
669                     >
670             ]
671             )
```

```
672                 oPushCurTokenMark                                   121
673                 oPushCopyPos0                                       133
674                 @StatementList                                       *
675             | * :
676                 oPushCopyMark0                                      135
677                 oPushCopyPos0                                       133
678                 @Statement                                           *
679         ]
680         oPopPosAndMark                                              123
681         oPopPosAndMark ;                                            123
682
683 % end of ClassDeclaration
684
685
686
687
688 ProcedureDeclaration :                                               *
689
690         'procedure'
691         oPushCurTokenMark                                            83   121
692         oSetupPos0                                                  138
693         oCcompress1                                                 141
694         oPushCopyMark0                                              135
695         oPushCopyPos0                                               133
696         @DeclarationPart                                             *
697         oCcompress0                                                 140
698         'semicolon'
699         [oChooseNextInputToken                                       92   153
700             |'begin' :                                               26
701                 oBlankLine                                          127
702             | * :
703         ]
704         oPushCurTokenMark                                           121
705         oMoveCmnts                                                  126
706         oSetupPos0                                                  138
707         {[=
708             |'value', 'name' :                                      104    72
709                 oPushCurTokenMark                                   121
710                 oSetupIndentPos0                                    137
711                 oCcompress1                                         141
712                 oPushCurTokenMark                                   121
713                 oPushIndentPos0                                     131
714                 @IdentList                                           *
715                 oCcompress0                                         140
716                 oPopMark                                            125
717                 oCcompress0                                         140
718                 oSwapMarks                                          128
719                 oPopMark                                            125
720                 'semicolon'                                          92
721                 oPushCurTokenMark                                   121
722                 oMoveCmnts                                          126
723                 [oChooseNextInputToken                              153
724                     |'begin' :                                       26
725                         oBlankLine                                  127
726                     | * :
727                 ]
```

```
                oSetupPos0                                          138
            | * :
                oSwapMarks                                          128
                oPopMark                                            125
                >
         ]
        )
        {[oChooseCurInputToken                                      152
           |'integer','real','boolean','character','text',           64    87    28    29    97
            'short','long','ref' :                                   93    69    88
                oPushCopyMark0                                      135
                oPushIndentPos0                                     131
                @Type                                                *
                oCcompress0                                         140
                oPushCurTokenMark                                   121
                oPushIndentPos0                                     131
                [=
                   |'array', 'procedure' :                           21    83
                       oSetupPos0                                   138
                       oCcompress1                                  141
                   | * :
                 ]
                oPushCopyMark0                                      135
                oPushIndentPos0                                     131
                @IdentList                                           *
                oCcompress0                                         140
                oPopPosAndMark                                      123
                oCcompress0                                         140
                'semicolon'                                          92
                oPopMark                                            125
                oPushCurTokenMark                                   121
                oMoveCmnts                                          126
                [oChooseNextInputToken                              153
                   |'begin' :                                        26
                        oBlankLine                                  127
                   | * :
                ]
                oSetupPos0                                          138
           |'array','procedure','label','switch' :                   21    83    66    96
                [=
                   |'array','procedure','label','switch':           21    83    66    96
                ]
                oPushCurTokenMark                                   121
                oPushIndentPos0                                     131
                oSetupPos0                                          138
                oCcompress1                                         141
                oPushCopyMark0                                      135
                oPushIndentPos0                                     131
                @IdentList                                           *
                oCcompress0                                         140
                oPopPosAndMark                                      123
                oCcompress0                                         140
                'semicolon'                                          92
                oPopMark                                            125
                oPushCurTokenMark                                   121
                oMoveCmnts                                          126
```

```
784                  [oChooseNextInputToken                                153
785                     |'begin' :                                          26
786                          oBlankLine                                     127
787                     | * :
788                  ]
789                  oSetupPos0                                            138
790               | * :
791                    >
792            ]
793           }
794           oPushCopyMark0                                               135
795           oPushIndentPos0                                              131
796           @Statement                                                    *
797           oCcompress0                                                  140
798           oPopMark                                                     125
799           oPopPosAndMark ;                                             123
800
801 % end of ProcedureDeclaration
802
803
804
805
806 StatementList :                                                        *
807
808           oPushCopyMark0                                               135
809           oPushIndentPos0                                              131
810           @Statement                                                    *
811           {[=
812              |'semicolon' :                                             92
813                   oPushCurTokenMark                                    121
814                   oMoveCmnts                                           126
815                   [oChooseNextInputToken                               153
816                      |'begin' :                                         26
817                           oBlankLine                                   127
818                      | * :
819                   ]
820                   oSetupPos0                                           138
821                   oPushCopyMark0                                       135
822                   oPushIndentPos0                                      131
823                   @Statement                                            *
824                   oCcompress0                                          140
825                   oPopMark                                             125
826                   [=
827                      |'inner' :                                         63
828                           oSetupGCurPosNc                              119
829                      | * :
830                   ]
831              |'end' :                                                   46
832                   oPushCurTokenMark                                    121
833                   oSetupPos0                                           138
834                   {[oChooseCurInputToken                               152
835                      |'semicolon','else','when','end','endofFile' :     92    45   106    46    49
836                           >
837                      | * :
838                           oMatchCmnt                                   129
839                           oPushCurTokenMark                            121
```

```
840                         oSetupPos0                                  138
841                         oCcompress1                                 141
842                         oSwapMarks                                  128
843                         oPopMark                                    125
844                     ]
845                     }
846                     oPopMark                                        125
847                     >
848                | * :
849                     >
850             ]
851             }
852             oPopPosAndMark ;                                        123
853
854 % end of StatementList
855
856
857
858
859 ExpressionList :                                                     *
860
861             oPushCopyMark0                                          135
862             oPushCopyMark0                                          135
863             oPushIndentPos0                                         131
864             @Expression                                              *
865             oCcompress1                                             141
866             {[=
867                |'comma', 'colon' :                                    33    32
868                     oPushCurTokenMark                              121
869                     oMoveCmnts                                     126
870                     oSetupPos0                                     138
871                     oPushCopyMark0                                 135
872                     oPushIndentPos0                                131
873                     @Expression                                     *
874                     oCcompress0                                    140
875                     oCcompress1                                    141
876                     oSwapMarks                                     128
877                     oPopMark                                       125
878                | * :
879                     >
880             ]
881             }
882             oPopMark                                               125
883             oPopPosAndMark ;                                       123
884
885 % end of ExpressionList
886
887
888
889
890 IdentList :                                                          *
891
892             oPushCopyMark0                                          135
893             oPushCopyMark0                                          135
894             {[=
895                |'ident' :                                            58
```

```
896                 oSetupPos0                                  138
897                 oCcompress0                                 140
898                 oCcompress1                                 141
899                 oSwapMarks                                  128
900                 oPopMark                                    125
901                 [=
902                    |'comma', 'colon' :                       33    32
903                         oPushCurTokenMark                   121
904                         oMoveCmnts                          126
905                         oSetupPos0                          138
906                    | * :
907                         >
908                 ]
909              | *:
910                 oPopMark                                    125
911                 >
912           ]
913         }
914         oPopMark                                            125
915         oPopPosAndMark ;                                    123
916
917 % end of IdentList
918
919
920
921
922 DeclarationPart :                                            *
923
924         'ident'
925         oPushCurTokenMark                                    58   121
926         oPushIndentPos0                                     131
927         oSetupPos0                                          138
928         oCcompress1                                         141
929         'leftParen'                                          68
930         oPushCurTokenMark                                   121
931         oPushIndentPos0                                     131
932         oSetupPos0                                          138
933         oPushCopyMark0                                      135
934         oPushCopyPos0                                       133
935         @ExpressionList                                     859
936         oCcompress0                                         140
937         'rightParen'                                         91
938         oSetupPos0                                          138
939         oCcompress0                                         140
940         oPopPosAndMark                                      123
941         oCcompress0                                         140
942         oPopPosAndMark                                      123
943         oPopPosAndMark ;                                    123
944
945 % end of DeclarationPart
946
947
948
949
950 Declaration :                                                *
951
```

```
952         [oChooseCurInputToken                                          152
953            |'integer','real','boolean','character','text',              64    87    28    29    97
954             'short','long','ref':                                       93    69    88
955                oPushCopyMark0                                          135
956                oPushIndentPos0                                         131
957                @Type                                                     *
958                oPushCopyMark0                                          135
959                oPushIndentPos0                                         131
960                @IdentList                                              890
961                oCcompress0                                             140
962            | * :
963         ]
964         [oChooseCurInputToken                                          152
965            |'array' :                                                   21
966                oPushCopyMark0                                          135
967                oPushIndentPos0                                         131
968                @ArrayDeclaration                                         *
969            |'switch' :                                                  96
970                oPushCopyMark0                                          135
971                oPushIndentPos0                                         131
972                @SwitchDeclaration                                        *
973            |'procedure' :                                               83
974                oPushCopyMark0                                          135
975                oPushIndentPos0                                         131
976                @ProcedureDeclaration                                   688
977            |'class' :                                                   31
978                oPushCopyMark0                                          135
979                oPushIndentPos0                                         131
980                @ClassDeclaration                                       430
981            |'ident' :                                                   58
982                [oChooseNextInputToken                                  153
983                   |'class' :                                           31
984                       oPushCopyMark0                                   135
985                       oPushIndentPos0                                  131
986                       @ClassDeclaration                                430
987                   | * :
988                ]
989            |'external' :                                                51
990                oPushCopyMark0                                          135
991                oPushIndentPos0                                         131
992                @ExternalDeclaration                                    233
993            | * :
994         ]
995         oPopPosAndMark ;                                               123
996
997 % end of Declaration
998
999
1000
1001
1002 ArrayList :                                                            *
1003
1004        oPushCopyMark0                                                  135
1005        oPushIndentPos0                                                 131
1006        @IdentList                                                      890
1007        'leftParen'                                                      68
```

```
1008          oPushCurTokenMark                                  121
1009          oPushIndentPos0                                    131
1010          oPushIndentPos0                                    131
1011          oSetupPos0                                         138
1012          oPushCopyMark0                                     135
1013          oPushGCurPosNc                                     120
1014          @ExpressionList                                    859
1015          oCcompress0                                        140
1016          'rightParen'                                        91
1017          oSetupPos0                                         138
1018          oCcompress0                                        140
1019          oPopPos                                            124
1020          oPopPosAndMark                                     123
1021          oPopPosAndMark ;                                   123
1022
1023  % end of ArrayList
1024
1025
1026
1027
1028  ArrayDeclaration :                                          *
1029
1030          'array'
1031          oPushCurTokenMark                                   21   121
1032          oSetupPos0                                         138
1033          oCcompress1                                        141
1034          oPushCopyMark0                                     135
1035          oPushIndentPos0                                    131
1036          @ArrayList                                        1002
1037          oCcompress0                                        140
1038          {[=
1039             |'comma' :                                       33
1040                  oPushCurTokenMark                          121
1041                  oMoveCmnts                                 126
1042                  oSetupIndentPos0                           137
1043                  oPushCopyMark0                             135
1044                  oPushIndentPos0                            131
1045                  @ArrayList                                1002
1046                  oCcompress0                                140
1047                  oCcompress1                                141
1048                  oSwapMarks                                 128
1049                  oPopMark                                   125
1050             | * :
1051                  >
1052           ]
1053          }
1054          oPopMark                                           125
1055          oPopPosAndMark ;                                   123
1056
1057  % end of ArrayDeclaration
1058
1059
1060
1061
1062  SwitchDeclaration :                                         *
1063
```

```
1064          'switch'
1065          oPushCurTokenMark                                  96   121
1066          oSetupPos0                                        138
1067          oCcompress1                                       141
1068          'ident'
1069          oPushCurTokenMark                                  53   121
1070          oSetupIndentPos0                                  137
1071          'becomes'
1072          oPushCurTokenMark                                  24   121
1073          oSetupIndentPos0                                  137
1074          oPushCopyMark0                                    135
1075          oPushIndentPos0                                   131
1076          @ExpressionList                                   859
1077          oCcompress0                                       140
1078          oPopMark                                          125
1079          oCcompress0                                       140
1080          oPopMark                                          125
1081          oCcompress0                                       140
1082          oPopMark                                          125
1083          oPopPosAndMark ;                                  123
1084
1085 % end of SwitchDeclaration
1086
1087
1088
1089
1090 Statement :                                                 *
1091
1092        [oChooseCurInputToken                               152
1093           |'ident' :                                        58
1094                [oChooseNextInputToken                      153
1095                   |'colon' :                               32
1096                        'ident'
1097                        oSetupPos0                           58   138
1098                        oCcompress0                         140
1099                        'colon'                              32
1100                        oSetupGCurPosNc                     119
1101                        oPushCopyMark0                      135
1102                        oPushCopyPos0                       133
1103                        @Statement                         1090
1104                   |'begin' :                               26
1105                        'ident'
1106                        oSetupPos0                           58   138
1107                        oCcompress0                         140
1108                        oPushCopyMark0                      135
1109                        oPushCopyPos0                       133
1110                        @UnconditionalStatement             324
1111                   | * :
1112                        oPushCurTokenMark                   121
1113                        oPushCopyPos0                       133
1114                        @StatementPart                        *
1115                ]
1116           |'inner' :                                        63
1117           | * :
1118                   oPushCopyMark0                           135
1119                   oPushCopyPos0                            133
```

```
1120                    @StatementPart                                  *
1121            ]
1122            oPopPosAndMark ;                                        123
1123
1124 % end of Statement
1125
1126
1127
1128
1129 StatementPart :                                                    *
1130
1131            [=
1132               |'if' :                                              59
1133                    oPushCurTokenMark                               121
1134                    oSetupPos0                                      138
1135                    oCcompress1                                     141
1136                    oPushCopyMark0                                  135
1137                    oPushIndentPos0                                 131
1138                    @Expression                                     *
1139                    oCcompress0                                     140
1140                    'then'
1141                    oPushCurTokenMark                               99    121
1142                    oSetupPos0                                      138
1143                    oPushCopyMark0                                  135
1144                    oPushCopyPos0                                   133
1145                    @Statement                                      1090
1146                    oCcompress0                                     140
1147                    oPopMark                                        125
1148                    [=
1149                       |'else' :                                    45
1150                            oPushCurTokenMark                       121
1151                            oSetupPos0                              138
1152                            oPushCopyMark0                          135
1153                            oPushIndentPos0                         131
1154                            @Statement                              1090
1155                            oCcompress0                             140
1156                            oPopMark                                125
1157                       | * :
1158                    ]
1159                    oCcompress0                                     140
1160               |'for' :                                             52
1161                    oPushCurTokenMark                               121
1162                    oSetupPos0                                      138
1163                    oCcompress1                                     141
1164                    'ident'
1165                    oPushCurTokenMark                               58    121
1166                    oPushIndentPos0                                 131
1167                    oSetupPos0                                      138
1168                    oCcompress1                                     141
1169                    [=
1170                       |'becomes' :                                 24
1171                       |'denotes' :                                 40
1172                    ]
1173                    oPushCurTokenMark                               121
1174                    oSetupPos0                                      138
1175                    oPushIndentPos0                                 131
```

```
1176                {
1177                 oPushCopyMark0                                  135
1178                 oPushCopyMark0                                  135
1179                 oPushCopyPos0                                   133
1180                 @Expression                                      *
1181                 oCcompress1                                     141
1182                 [=
1183                   |'step' :                                      95
1184                        oPushCurTokenMark                        121
1185                        oSetupPos0                               138
1186                        oPushCurTokenMark                        121
1187                        oPushIndentPos0                          131
1188                        @Expression                               *
1189                        oCcompress0                              140
1190                        'until'                                  103
1191                        oPushCurTokenMark                        121
1192                        oSetupPos0                               138
1193                        oPushCurTokenMark                        121
1194                        oPushIndentPos0                          131
1195                        @Expression                               *
1196                        oCcompress0                              140
1197                        oPopMark                                 125
1198                        oCcompress0                              140
1199                        oPopMark                                 125
1200                   |'while' :                                    107
1201                        oPushCurTokenMark                        121
1202                        oSetupPos0                               138
1203                        oPushCurTokenMark                        121
1204                        oPushIndentPos0                          131
1205                        @Expression                               *
1206                        oCcompress0                              140
1207                        oPopMark                                 125
1208                   | * :
1209                 ]
1210                 oCcompress0                                     140
1211                 oPopMark                                        125
1212                 oCcompress1                                     141
1213                 oSwapMarks                                      128
1214                 oPopMark                                        125
1215                 [=
1216                   |'comma' :                                     33
1217                        oPushCurTokenMark                        121
1218                        oMoveCmnts                               126
1219                        oSetupPos1                               139
1220                   | * :
1221                        oPopMark                                 125
1222                        >
1223                 ]
1224                }
1225                oPopPos                                         124
1226                oPopPos                                         124
1227                'do'
1228                oPushCurTokenMark                                42   121
1229                oSetupPos0                                      138
1230                oPushCopyMark0                                  135
1231                oPushCopyPos0                                   133
```

```
1232                @Statement                                  1090
1233                oCcompress0                                  140
1234                oPopMark                                     125
1235           |'inspect' :                                        63
1236                oPushCurTokenMark                            121
1237                oSetupPos0                                   138
1238                oCcompress1                                  141
1239                oPushCurTokenMark                            121
1240                oPushIndentPos0                              131
1241                @Expression                                    *
1242                oCcompress0                                  140
1243                [=
1244                   |'do' :                                    42
1245                        oPushCurTokenMark                    121
1246                        oSetupPos0                           138
1247                        oCcompress0                          140
1248                        oPushCopyMark0                       135
1249                        oPushCopyPos0                        133
1250                        @Statement                          1090
1251                        oCcompress0                          140
1252                        oPopMark                             125
1253                   | * :
1254                        {[=
1255                           |'when' :                         106
1256                                oPushCurTokenMark           121
1257                                oSetupPos0                  138
1258                                'ident'
1259                                oSetupIndentPos0             58   137
1260                                oCcompress0                 140
1261                                'do'
1262                                oSetupPos0                   42   138
1263                                oCcompress0                 140
1264                                oPushCopyMark0              135
1265                                oPushCopyPos0               133
1266                                @Statement                 1090
1267                                oCcompress0                 140
1268                                oPopMark                    125
1269                             | * :
1270                                >
1271                         ]
1272                        }
1273                        [=
1274                          |'otherwise' :                     81
1275                                oPushCurTokenMark           121
1276                                oSetupPos0                  138
1277                                oPushCopyMark0              135
1278                                oPushIndentPos0             131
1279                                @Statement                 1090
1280                                oCcompress0                 140
1281                                oPopMark                    125
1282                             | * :
1283                         ]
1284           ]
1285      |'while' :                                             107
1286           oPushCurTokenMark                                 121
1287           oSetupPos0                                        138
```

```
1288                oCcompress1                                  141
1289                oPushCurTokenMark                            121
1290                oPushIndentPos0                              131
1291                @Expression                                    *
1292                oCcompress0                                  140
1293                'do'
1294                oPushCurTokenMark                             42   121
1295                oSetupPos0                                   138
1296                oPushCopyMark0                               135
1297                oPushCopyPos0                                133
1298                @Statement                                  1090
1299                oCcompress0                                  140
1300             | * :
1301                oPushCopyMark0                               135
1302                oPushCopyPos0                                133
1303                @UnconditionalStatement                      324
1304                oPushCurTokenMark                            121
1305          ]
1306          oPopMark                                           125
1307          oPopPosAndMark ;                                   123
1308
1309 % end of StatementPart
1310
1311
1312
1313
1314 Type :                                                        *
1315          [=
1316             |'integer', 'real', 'boolean', 'character', 'text' :   64    87    28    29    97
1317                 oSetupPos0                                   138
1318             |'short' :                                        93
1319                 oSetupPos0                                   138
1320                 'integer'
1321                 oSetupGCurPosNc                              64   119
1322             |'long' :                                         69
1323                 oSetupPos0                                   138
1324                 'real'
1325                 oSetupGCurPosNc                              87   119
1326             |'ref' :                                          88
1327                 oPushCurTokenMark                            121
1328                 oSetupPos0                                   138
1329                 oPushIndentPos0                              131
1330                 'leftParen'                                   63
1331                 oPushCurTokenMark                            121
1332                 oSetupPos0                                   138
1333                 'ident'
1334                 oSetupPos0                                    58   138
1335                 oCcompress0                                  140
1336                 'rightParen'                                  91
1337                 oSetupPos0                                   138
1338                 oCcompress0                                  140
1339                 oPopPosAndMark                               123
1340                 oCcompress0                                  140
1341                 oPopMark                                     125
1342             | * :
1343          ]
```

```
1344          oPopPosAndMark ;                                  123
1345
1346  % end of Type
1347
1348
1349
1350
1351  String :                                                  *
1352
1353          oPushCurTokenMark                                 121
1354          {[=
1355             |'simpleString' :                               94
1356                  oPushCurTokenMark                         121
1357                  oSetupPos0                                138
1358                  oCcompress1                               141
1359                  oSwapMarks                                128
1360                  oPopMark                                  125
1361             | * :
1362                  >
1363           ]
1364          }
1365          oPopMark                                          125
1366          oPopPosAndMark ;                                  123
1367
1368  % end of String
1369
1370
1371
1372
1373  Expression :                                              *
1374
1375          [=
1376             |'if' :                                         59
1377                  oPushCurTokenMark                         121
1378                  oSetupPos0                                138
1379                  oCcompress1                               141
1380                  oPushCurTokenMark                         121
1381                  oPushIndentPos0                           131
1382                  @Expression                              1373
1383                  oCcompress0                               140
1384                  'then'
1385                  oPushCurTokenMark                          99   121
1386                  oSetupPos0                                138
1387                  oPushCopyMark0                            135
1388                  oPushCopyPos0                             133
1389                  @SimpleExpression                           *
1390                  oCcompress0                               140
1391                  oPopMark                                  125
1392                  'else'
1393                  oPushCurTokenMark                          45   121
1394                  oSetupPos0                                138
1395                  oPushCopyMark0                            135
1396                  oPushIndentPos0                           131
1397                  @Expression                              1373
1398                  oCcompress0                               140
1399                  oPopMark                                  125
```

```
1400                oCcompress0                                   140
1401                oPopMark                                      125
1402             | * :
1403                oPushCopyMark0                                135
1404                oPushCopyPos0                                 133
1405                @SimpleExpression                               *
1406          ]
1407          oPopPosAndMark ;                                    123
1408
1409 % end of Expression
1410
1411
1412
1413
1414 SimpleExpression :                                             *
1415
1416          oPushCopyMark0                                      135
1417          oPushCopyPos0                                       133
1418          @Implication                                          *
1419          [=
1420             |'eqv' :                                          49
1421                oPushCurTokenMark                             121
1422                oSetupPos0                                    138
1423                oPushCurTokenMark                             121
1424                oPushGCurPosNc                                120
1425                @Implication                                    *
1426                oCcompress0                                   140
1427                oPopMark                                      125
1428             | * :
1429          ]
1430          oCcompress0                                         140
1431          oPopPosAndMark ;                                    123
1432
1433 % end of SimpleExpression
1434
1435
1436
1437
1438 Implication :                                                 *
1439
1440          oPushCopyMark0                                      135
1441          oPushCopyPos0                                       133
1442          @Disjunction                                          *
1443          [=
1444             |'imp' :                                          60
1445                oPushCurTokenMark                             121
1446                oSetupPos0                                    138
1447                oPushCurTokenMark                             121
1448                oPushGCurPosNc                                120
1449                @Disjunction                                    *
1450                oCcompress0                                   140
1451                oPopMark                                      125
1452             | * :
1453          ]
1454          oCcompress0                                         140
1455          oPopPosAndMark ;                                    123
```

```
1456
1457 % end of Implication
1458
1459
1460
1461
1462 Disjunction :                                                    *
1463
1464         oPushCopyMark0                                          135
1465         oPushCopyPos0                                           133
1466         @Conjunction                                             *
1467         {[=
1468            |'or' :                                              79
1469                 oPushCurTokenMark                              121
1470                 oSetupPos0                                     138
1471                 oPushCurTokenMark                              121
1472                 oPushGCurPosNc                                 120
1473                 @Conjunction                                    *
1474                 oCcompress0                                    140
1475                 oPopMark                                       125
1476            | * :
1477                 oCcompress0                                    140
1478                 >
1479          ]
1480         }
1481         oPopPosAndMark ;                                        123
1482
1483 % end of Disjuction
1484
1485
1486
1487
1488 Conjunction :                                                    *
1489
1490         oPushCopyMark0                                          135
1491         oPushCopyPos0                                           133
1492         @Negation                                                *
1493         {[=
1494            |'and' :                                             20
1495                 oPushCurTokenMark                              121
1496                 oSetupPos0                                     138
1497                 oPushCurTokenMark                              121
1498                 oPushGCurPosNc                                 120
1499                 @Negation                                       *
1500                 oCcompress0                                    140
1501                 oPopMark                                       125
1502            | * :
1503                 oCcompress0                                    140
1504                 >
1505          ]
1506         }
1507         oPopPosAndMark ;                                        123
1508
1509 % end of Conjunction
1510
1511
```

```
1512
1513
1514 Negation :                                                          *
1515
1516         [=
1517            |'not' :                                                 76
1518                 oPushCurTokenMark                                  121
1519                 oSetupPos0                                         138
1520                 oCcompress1                                        141
1521                 oPushCurTokenMark                                  121
1522                 oPushIndentPos0                                    131
1523                 @Relation                                           *
1524                 oCcompress0                                        140
1525                 oPopMark                                           125
1526            | * :
1527                 oPushCopyMark0                                     135
1528                 oPushCopyPos0                                      133
1529                 @Relation                                           *
1530         ]
1531         oPopPosAndMark ;                                           123
1532
1533 % end of Negation
1534
1535
1536
1537
1538 Relation :                                                          *
1539
1540         oPushCopyMark0                                             135
1541         oPushCopyPos0                                              133
1542         @Sum                                                        *
1543         oCcompress0                                                140
1544         [=
1545            |'eq', 'ge', 'gt' ,'le',                                 47    54    55    67
1546             'lt', 'ne' , 'refEqual', 'refNotequal', 'in', 'is' :    70    73    89    90    61    65
1547                 oPushCurTokenMark                                  121
1548                 oSetupPos0                                         138
1549                 oPushCopyMark0                                     135
1550                 oPushGCurPosNc                                     120
1551                 @Sum                                                *
1552                 oCcompress0                                        140
1553                 oPopMark                                           125
1554            | * :
1555         ]
1556         oCcompress0                                                140
1557         oPopPosAndMark ;                                           123
1558
1559 % end of Relation
1560
1561
1562
1563
1564 Sum :                                                              *
1565
1566         [=
1567            |'plus', 'minus' :                                       82    71
```

```
1568                    oPushCurTokenMark                                121
1569                    oSetupPos0                                       138
1570                 | * :
1571                    oPushCopyMark0                                   135
1572              ]
1573              oPushCopyMark0                                         135
1574              oPushCopyPos0                                          133
1575              @Factor                                                 *
1576              oCcompress1                                            141
1577              oPopMark                                               125
1578              oPushCurTokenMark                                      121
1579              oPushCopyPos0                                          133
1580              @PartialSum                                             *
1581              oPopPosAndMark ;                                       123
1582
1583 % end of Sum
1584
1585
1586
1587
1588 PartialSum :                                                        *
1589
1590              oPushCopyPos0                                          133
1591              oPushCopyMark0                                         135
1592              @PartialTerm                                            *
1593              oPushCopyMark0                                         135
1594              {[=
1595                 |'plus', 'minus' :                                   82     71
1596                    oPushCurTokenMark                                121
1597                    oSetupPos0                                       138
1598                    oPushCopyMark0                                   135
1599                    oPushGCurPosNc                                   120
1600                    @Term                                            *
1601                    oCcompress0                                      140
1602                    oCcompress1                                      141
1603                    oSwapMarks                                       128
1604                    oPopMark                                         125
1605                 | * :
1606                    oCcompress0                                      140
1607                    >
1608               ]
1609              }
1610              oPopMark                                               125
1611              oPopPosAndMark ;                                       123
1612
1613 % end of PartialSum
1614
1615
1616
1617
1618 Term :                                                              *
1619
1620              oPushCopyMark0                                         135
1621              oPushCopyPos0                                          133
1622              @Factor                                                 *
1623              oCcompress0                                            140
```

```
1624          oPushCurTokenMark                                  121
1625          oPushCopyPos0                                      133
1626          @PartialTerm                                        *
1627          oPopPosAndMark ;                                   123
1628
1629  % end of Term
1630
1631
1632
1633
1634  PartialTerm :                                               *
1635
1636          oPushCopyMark0                                     135
1637          {[=
1638             |'times', 'divide' :                            100    41
1639                  oPushCurTokenMark                          121
1640                  oSetupPos0                                 138
1641                  oPushCopyMark0                             135
1642                  oPushGCurPosNc                             120
1643                  @Factor                                     *
1644                  oCcompress0                                140
1645                  oCcompress1                                141
1646                  oSwapMarks                                 128
1647                  oPopMark                                   125
1648             | * :
1649                  oCcompress0                                140
1650                  >
1651           ]
1652          }
1653          oPopMark                                           125
1654          oPopPosAndMark ;                                   123
1655
1656  % end of PartialTerm
1657
1658
1659
1660
1661  Factor :                                                    *
1662
1663          oPushCopyMark0                                     135
1664          oPushCopyPos0                                      133
1665          @Variable                                           *
1666          oCcompress0                                        140
1667          oPushCurTokenMark                                  121
1668          oPushCopyPos0                                      133
1669          @PartialFactor                                      *
1670          oPopPosAndMark ;                                   123
1671
1672  % end of Factor
1673
1674
1675
1676
1677  PartialFactor :                                             *
1678
1679          {[=
```

```
1680            |'expon' :                                                  51
1681                 oPushCurTokenMark                                    121
1682                 oSetupPos0                                           138
1683                 oPushCopyMark0                                       135
1684                 oPushGCurPosNc                                       120
1685                 @Variable                                             *
1686                 oCcompress0                                          140
1687                 oPopMark                                             125
1688            | * :
1689                 oCcompress0                                          140
1690                 >
1691          ]
1692         }
1693         oPopPosAndMark ;                                             123
1694
1695 % end of PartialFactor
1696
1697
1698
1699
1700 Variable :                                                            *
1701
1702         oPushCurTokenMark                                            121
1703         [=
1704            |'ident', 'notext', 'none' , 'charsequence' :              58    77    76    30
1705                 oSetupPos0                                           138
1706                 oCcompress1                                          141
1707            |'number' :                                                78
1708                 oSetupPos0                                           138
1709                 oCcompress1                                          141
1710                 ([=
1711                    |'number' :                                       78
1712                         oPushCurTokenMark                            121
1713                         oSetupPos0                                   138
1714                         oCcompress1                                  141
1715                         oSwapMarks                                   128
1716                         oPopMark                                     125
1717                    | * :
1718                         >
1719                  ]
1720                 )
1721            |'new', 'this' :                                           74    99
1722                 oSetupPos0                                           138
1723                 oCcompress1                                          141
1724                 'ident'
1725                 oSetupIndentPos0                                      58   137
1726                 oCcompress0                                          140
1727            |'simpleString' :                                          94
1728                 oSetupPos0                                           138
1729                 oCcompress1                                          141
1730                 oPushCurTokenMark                                    121
1731                 oPushCopyPos0                                        133
1732                 @String                                             1351
1733                 oCcompress0                                          140
1734            |'leftParen' :                                             68
1735                 oSetupPos0                                           138
```

```
1736                oCcompress1                                  141
1737                oPushCopyMark0                               135
1738                oPushCopyPos0                                133
1739                @ExpressionList                              859
1740                oCcompress0                                  140
1741                'rightParen'
1742                oSetupPos0                                    91   138
1743                oCcompress0                                  140
1744             | * :
1745          ]
1746          oPushIndentPos0                                     131
1747          {[=
1748             |'dot', 'qua' :                                   43    85
1749                oPushCurTokenMark                            121
1750                oSetupPos0                                   138
1751                [=
1752                   |'ident' :                                 58
1753                      oSetupPos0                             138
1754                      oCcompress0                            140
1755                      oCcompress1                            141
1756                      oSwapMarks                             128
1757                      oPopMark                               125
1758                   |'number' :                                78
1759                      oSetupPos0                             138
1760                      oCcompress0                            140
1761                      oCcompress1                            141
1762                      oSwapMarks                             128
1763                      oPopMark                               125
1764                      {[=
1765                         |'number' :                          78
1766                            oPushCurTokenMark                121
1767                            oSetupPos0                       138
1768                            oCcompress1                      141
1769                            oSwapMarks                       128
1770                            oPopMark                         125
1771                         | * :
1772                            >
1773                       ]
1774                      }
1775                ]
1776             |'basisOfTen' :                                   24
1777                oSetupPos0                                   138
1778                [=
1779                   |'plus','minus' :                          82    71
1780                      oSetupPos0                             138
1781                   | * :
1782                ]
1783                'number'
1784                oSetupPos0                                    78   138
1785             |'leftParen' :                                    69
1786                oPushCurTokenMark                            121
1787                oSetupPos0                                   138
1788                oPushCopyMark0                               135
1789                oPushCopyPos0                                133
1790                @ExpressionList                              859
1791                oCcompress0                                  140
```

```
1792                'rightParen'
1793                oSetupPos0                                  91   138
1794                oCcompress0                                140
1795                oCcompress1                                141
1796                oSwapMarks                                 128
1797                oPopMark                                   125
1798                oCcompress0                                140
1799             | * :
1800                >
1801          ]
1802         }
1803         oCcompress0                                       140
1804         oPopPosAndMark                                    123
1805         oPopPosAndMark ;                                  123
1806
1807 % end of Variable
1808
1809
1810 _|_                                                        *     *
1811
```

cross reference * is def = is asg

```
-'-
 'activate'              18*  373
 'after'                 19*  382
 'and'                   20* 1494
 'array'                 21*  495   516   518   596   617   619   745   766   768   965  1031
 'at'                    22*  382
 'basisOfTen'            24* 1776
 'becomes'               24*  407  1072  1170
 'before'                25*  382
 'begin'                 26*  196   209   327   452   476   511   535   568   612   636   651   700   724   761   785
                        816  1104
 'boolean'               28*  486   587   736   953  1316
 'character'             29*  486   587   736   953  1316
 'charsequence'          30* 1704
 'class'                 31*  213   241   441   977   983
 'colon'                 32*  584   867   902  1095  1099
 'comma'                 33*  286   867   902  1039  1216
 'comment'               34*
 'delay'                 39*  382
 'denotes'               40*  407  1171
 'divide'                41* 1638
 'do'                    42* 1228  1244  1262  1294
 'dot'                   43* 1748
 'else'                  45*  835  1149  1393
 'end'                   46*  831   835
 'endofFile'             49*  173   835
 'eq'                    47*  268  1545
 'eqv'                   49* 1420
 'expon'                 51* 1680
 'external'              51*  236   989
 'for'                   52* 1160
 'ge'                    54* 1545
 'go'                    56*  362
 'goto'                  57*  353
 'gt'                    55* 1545
 'hidden'                57*  547   551
 'ident'                 56*  245   265   433   895   925   981  1069  1093  1097  1106  1165  1259  1334  1704  1725
                       1752
 'if'                    59* 1132  1376
 'imp'                   60* 1444
 'in'                    61* 1546
 'inner'                 63*  827  1116
 'inspect'               63* 1235
 'integer'               64*  486   587   736   953  1316  1321
 'is'                    55*  297  1546
 'label'                 66*  516   518   617   619   766   768
 'le'                    67* 1545
 'leftParen'             68*  929  1007  1330  1734  1785
 'long'                  69*  487   588   737   954  1322
 'lt'                    70* 1546
 'minus'                 71* 1567  1595  1779
 'name'                  72*  709
 'ne'                    73* 1546
```

```
                      508   520   532   565   584   593   609   621   633   654   659   672   691   704   709   712
                      721   742   758   770   782   813   832   839   868   903   925   930  1008  1031  1040  1065
                     1069  1072  1112  1133  1141  1150  1161  1165  1173  1184  1186  1191  1193  1201  1203  1217
                     1228  1236  1239  1245  1256  1275  1286  1289  1294  1304  1327  1331  1353  1356  1377  1380
                     1385  1393  1421  1423  1445  1447  1469  1471  1495  1497  1518  1521  1547  1568  1578  1596
                     1624  1639  1667  1681  1702  1712  1730  1749  1766  1786
 oPushGCurPosNc       120*  1013  1424  1448  1472  1498  1550  1599  1642  1684
 oPushIndentPos0      131*   176   250   259   260   301   305   358   369   378   386   464   489   493   501   521
                      525   557   558   590   594   602   622   626   649   713   739   743   751   771   775   795
                      809   822   863   872   926   931   956   959   967   971   975   979   985   991  1005  1009
                     1010  1035  1044  1075  1137  1153  1166  1175  1187  1194  1204  1240  1278  1290  1329  1381
                     1396  1522  1746
 oPushIndentPos1      132*
 oPushNextTokenMark   122*
 oSetupGCurPosNc      119*   242   367   828  1100  1321  1325
 oSetupIndentPos0     137*   246   462   548   552   710  1042  1070  1073  1259  1725
 oSetupPos0           138*   238   257   266   270   299   329   339   355   364   375   384   393   409   435   442
                      458   480   496   515   522   539   572   581   595   597   616   623   640   652   662   692
                      706   728   746   765   772   789   820   833   840   870   896   905   927   932   938  1011
                     1017  1032  1066  1097  1106  1134  1142  1151  1162  1167  1174  1185  1192  1202  1229  1237
                     1246  1257  1262  1276  1287  1295  1317  1319  1323  1328  1332  1334  1337  1357  1378  1386
                     1394  1422  1446  1470  1496  1519  1548  1569  1597  1640  1682  1705  1708  1713  1722  1728
                     1735  1742  1750  1753  1759  1767  1777  1780  1784  1787  1793
 oSetupPos1           139*   200   239  1219
 oSwapMarks           128*   283   418   444   482   562   577   645   718   730   842   876   899  1048  1213  1359
                     1603  1646  1715  1756  1762  1769  1796

-P-
 PartialFactor       1669  1677*
 PartialSum          1580  1588*
 PartialTerm         1592  1626  1634*
 ProcedureDeclaration 223   306   698*  976
 Programs             168*

-R-
 Relation            1523  1529  1538*

-S-
 SimpleExpression    1389  1405  1414*
 SourceModule         177   186*
 Statement            678   796   810   823  1090* 1103  1145  1154  1232  1250  1266  1279  1298
 StatementList        351   674   806*
 StatementPart       1114  1120  1129*
 String               273  1351* 1732
 Sum                 1542  1551  1564*
 SwitchDeclaration    972  1062*

-T-
 tactivate             18*
 tafter                19*
 tand                  20*
 tarray                21*
 tat                   22*
 tbasisOfTen           23*
 tbecomes              24*
 tbefore               25*
```

```
 tplus                    81*
 tprior                   82*
 tprocedure               83*
 tprotected               84*
 tqua                     85*
 treactivate              86*
 treal                    87*
 tref                     88*
 trefEqual                89*
 trefNotequal             90*
 trightParen              91*
 tsemicolon               92*
 tshort                   93*
 tsimpleString            94*
 tstep                    95*
 tswitch                  96*
 ttext                    97*
 tthen                    98*
 tthis                    99*
 ttimes                  100*
 tto                     101*
 tunknown                102*
 tuntil                  103*
 tvalue                  104*
 tvirtual                105*
 twhen                   106*
 twhile                  107*
 txor                    108*
 Type                    220   252   302   490   591   740   957  1314*

-U-
 UnconditionalStatement  212   324* 1110  1303

-V-
 Variable               1665  1685  1700*

-_-
 -                       156*  156* 1810* 1810*

end exref  235 identifiers  1428 total references
 357 collisions.
```

Appendix 3

Input sample Simula code

```
EXTERNAL CLASS SIMULATION;   ! (sport)tst-sim-five:sim  ;
SIMULATION
BEGIN   COMMENT  ------  TEST PROCESS, ACTIVATION STATEMENTS ----
                 ------       IDLE,TERMINATED,TIME.          -----;

COMMENT ***************************************************************
         **********************                    ***********************
         **********************     TEST PW 21     ***********************
         **********************                    ***********************;

PROCESS CLASS P(I); INTEGER I;
         BEGIN OUTTEXT("P("); OUTINT (I,2);
              OUTTEXT(") ACTIVATED AT TIME = ");
              OUTFIX(TIME,3,8); OUTIMAGE; PASSIVE(I):=FALSE;
              ACTIVE(I):=TRUE; OUTSTATE(THIS P);
              ACTIVE(I):=FALSE; TERMINATD(I):=TRUE; PASSIVE(I):=FALSE;
         END;
PROCEDURE STARTUP;
           BEGIN OUTLINE("START UP PROCESSES:");
                OUTSTATE; REACTIVATE MAIN AT TIME+20.0; END;
PROCEDURE OUTSTATE(X); REF(P)X;
BEGIN COMMENT OUTPUT STATE OF PROCESS OBJECTS AND SQS;
     OUTTEXT("PROCESSES:          1          2          3          4");
      OUTTEXT("          5          6          7          8");
      OUTTEXT("          9          10");  OUTIMAGE;
      OUTTEXT("STATE:      ");
      FOR I:=1 STEP 1 UNTIL 10 DO
           BEGIN OUTTEXT("         ");
                IF PASSIVE(I) THEN OUTCHAR('P') ELSE OUTCHAR(' ');
                IF ACTIVE(I) THEN OUTCHAR('A')  ELSE OUTCHAR(' ');
                IF TERMINATD(I)  THEN OUTCHAR('T') ELSE OUTCHAR(' ');
           END; OUTIMAGE; OUTIMAGE;
      OUTTEXT("SQS:        ");
      FOR I:=1 STEP 1 UNTIL 10 DO
           BEGIN IF PA(I).IDLE THEN
                   BEGIN IF PA(I).TERMINATED THEN
                                 OUTTEXT("          T")
                             ELSE OUTTEXT("***ERROR***");
                   END ELSE
                   BEGIN REAL R; R:=PA(I).EVTIME;
                         IF I=X.I AND
                            ABS(R-TIME)>0.000001 OR PA(I).TERMINATED
                            THEN OUTTEXT("ERR") ELSE OUTTEXT("   ");
                         OUTFIX(R,3,8);
                   END;
           END;
      OUTIMAGE; OUTIMAGE; OUTIMAGE;
END OF OUTSTATE;
PROCEDURE GREATEOBJECTS;
         BEGIN   FOR I:=1 STEP 1 UNTIL 10 DO PA(I):-NEW P(I); END;
REAL PROCEDURE GETIME;
BEGIN GETIME:=ACTIME.GETREAL;
      ACTIME:-ACTIME.SUB(ACTIME.POS,ACTIME.LENGTH-ACTIME.POS+1);
END;
PROCEDURE OUTSTARS(T); VALUE T; TEXT T;
           BEGIN  OUTTEXT("****************************************");
                 OUTTEXT(T);
                 OUTTEXT("****************************************");
                 OUTIMAGE; OUTIMAGE;
           END;
PROCEDURE OUTLINE(T); VALUE T; TEXT T;
           BEGIN  OUTTEXT(T); OUTIMAGE; OUTIMAGE; END;
END;
```

Appendix 4

Output sample Simula code

```
  EXTERNAL CLASS SIMULATION  ! (sport)tst-sim-five.sim  ;
; SIMULATION
  BEGIN  COMMENT  ------  TEST PROCESS, ACTIVATION STATEMENTS ----
                  ------        IDLE,TERMINATED,TIME.          -----;
COMMENT ************************************************************
         **********************                    ************************
         **********************     TEST PW 21     ************************
         **********************                    ************************;
    PROCESS CLASS P ( I )
    ; INTEGER I

    ; BEGIN
        OUTTEXT ( "P(" )
      ; OUTINT ( I , 2 )
      ; OUTTEXT ( ") ACTIVATED AT TIME = " )
      ; OUTFIX ( TIME , 3 , 8 )
      ; OUTIMAGE
      ; PASSIVE ( I ) := FALSE
      ; ACTIVE ( I ) := TRUE
      ; OUTSTATE ( THIS P )
      ; ACTIVE ( I ) := FALSE
      ; TERMINATD ( I ) := TRUE
      ; PASSIVE ( I ) := FALSE
      ;
      END

  ; PROCEDURE STARTUP

    ; BEGIN
        OUTLINE ( "START UP PROCESSES:" )
      ; OUTSTATE
      ; REACTIVATE MAIN AT TIME + 20.0
      ;
      END

  ; PROCEDURE OUTSTATE ( X )
    ; REF ( P ) X

    ; BEGIN COMMENT OUTPUT STATE OF PROCESS OBJECTS AND SQS;
        OUTTEXT
          ( "PROCESSES:          1          2          3          4"
          )
      ; OUTTEXT ( "          5          6          7          8" )
      ; OUTTEXT ( "          9          10" )
      ; OUTIMAGE
      ; OUTTEXT ( "STATE:      " )
      ; FOR I := 1 STEP 1 UNTIL 10
        DO BEGIN
          OUTTEXT ( "         " )
        ; IF PASSIVE ( I ) THEN OUTCHAR ( 'P' ) ELSE OUTCHAR ( ' ' )
        ; IF ACTIVE ( I ) THEN OUTCHAR ( 'A' ) ELSE OUTCHAR ( ' ' )
        ; IF TERMINATD ( I ) THEN OUTCHAR ( 'T' ) ELSE OUTCHAR ( ' ' )
        ;
        END
      ; OUTIMAGE
      ; OUTIMAGE
      ; OUTTEXT ( "SQS:        " )
      ; FOR I := 1 STEP 1 UNTIL 10
        DO BEGIN
          IF PA ( I ) . IDLE
          THEN BEGIN
            IF PA ( I ) . TERMINATED
            THEN OUTTEXT ( "          T" )
            ELSE OUTTEXT ( "***ERROR***" )
        ;
        END
```

```
            ELSE BEGIN
                REAL R

              ; R := PA ( I ) . EVTIME
              ; IF I = X . I AND ABS ( R - TIME ) > 0.000001
                  OR PA ( I ) . TERMINATED
                THEN OUTTEXT ( "ERR" )
                ELSE OUTTEXT ( "   " )
              ; OUTFIX ( R , 3 , 8 )
              ;
              END
          ;
          END
        ; OUTIMAGE
        ; OUTIMAGE
        ; OUTIMAGE
        ;
        END OF OUTSTATE

    ; PROCEDURE CREATEOBJECTS

      ; BEGIN
          FOR I := 1 STEP 1 UNTIL 10
          DO PA ( I ) :- NEW P ( I )
        ;
        END

    ; REAL PROCEDURE GETIME

      ; BEGIN
          GETIME := ACTIME . GETREAL
        ; ACTIME
          :- ACTIME . SUB
            ( ACTIME . POS , ACTIME . LENGTH - ACTIME . POS + 1 )
        ;
        END

    ; PROCEDURE OUTSTARS ( T )
      ; VALUE T
      ; TEXT T

      ; BEGIN
          OUTTEXT ( "****************************************" )
        ; OUTTEXT ( T )
        ; OUTTEXT ( "****************************************" )
        ; OUTIMAGE
        ; OUTIMAGE
        ;
        END

    ; PROCEDURE OUTLINE ( T )
      ; VALUE T
      ; TEXT T

      ; BEGIN OUTTEXT ( T ) ; OUTIMAGE ; OUTIMAGE ; END

    ;
    END
;
```

SIMULA PRETTYPRINTER USING PASCAL

by

JUNG-JUIN CHEN

B.S., FU JEN CATHOLIC UNIVERSITY, 1979

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1986

ABSTRACT

Prettyprinter is a program designed to parse and reformat the whole text of a source program for a particular language. The advantages of a prettyprinter are that it saves programmers a lot of time editing programs and it enforces standard indentation conventions. This project is to design a prettyprinter program for the Simula language and make it available on the Interdata 8/32 machine under the Unix V7 operating system at Kansas State University's Computer Science Department.

The approach of this project is to use the language called Syntax/Semantic Language(S/SL) to write the parser and modify the existing Euclid prettyprinter(EPP) written in Pascal. S/SL is a pure control language without data or assignment, and data can only be manipulated via semantic operations. The implementation of S/SL program involved here is to translate the S/SL program into an S/SL table(or S-code) via an S/SL assembler, and then include this S-code in the Simula prettyprinter(SPP). Later, a program called S-code interpreter written in Pascal accesses the S-code and executes the functions of the S/SL program as a recursive descent parser. This project is divided into the following six phases :

1. The description of Simula's syntax by combining the syntax of Algol 60(the subset of Simula) and that of the extension part of Simula.
2. The implementation of the Syntax/Semantic Language(S/SL)

program for the parser of Simula.

3. The translation of the S/SL program into a S/SL table(or an S-code) via an S/SL assembler.
4. The modification of EPP's lexical scanner to make a new lexical scanner for Simula.
5. The replacement of the S/SL table in the EPP which is modified via step 3 by the S/SL table implemented for Simula to make a new Simula prettyprinter(SPP) program.
6. The implementation of the SPP.

The result generated by the above steps is a Simula prettyprinter written in Pascal. The main feature of the Simula prettyprinter is that it uses an S/SL program to serve as the recursive descent parser portion of the SPP. This implies that we can define the meaning of semantic mechanisms without directly being concerned with implementation details.