EUCLID PRETTY-PRINTER USING PASCAL

by

WUN-JEN LIN

BACHELOR OF BUSINESS, FENG-CHIA COLLEGE, 1976

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

1983

Approved by:

Rodney M. Bates
Major Professor

LD
2668
.R4
1983
L56
c.2

A11202 572193

Acknowledgements

I would like to express my sincere gratitude to my major advisor, Dr. Rodney Bates, for the guidance, assistance and great patience he provided me toward the completion of this report. Thanks are also given to Dr. Virgil Wallentine and Dr. William Hankley for serving on my committee. I also wish to express my appreciation to my sister, Hui-Ling, for her encouragement and for her careful typing of this report.

TABLE OF CONTENTS

Page

LIST OF FIGURES iii

Chapter

1. Introduction 1

   1.1 Background 1
   1.2 Purpose 1
   1.3 Approach 2
   1.4 Organization 3

2. Pretty-printing 4

   2.1 The motivation for pretty-printing 4
   2.2 The history of pretty-printers 4
   2.3 Functions of Euclid Pretty-Printer 5

3. Syntax/Semantic Language 13

   3.1 Syntax/Semantic Charts 13
   3.2 The concept of S/SL 17
   3.3 The features of S/SL 19
   3.4 Why S/SL is used in this project? 22
   3.5 The implementation of S/SL 23

4. Euclid Pretty-Printer 25

   4.1 The process overview 25
   4.2 The use of S/SL 28
   4.3 Modifications and insertions of PPP's semantic routines 32
   4.4 The modification of PPP's lexical scanner 34
   4.5 The translation and the inclusion of the S/SL program 35
   4.6 The inclusion of the S-code interpreter 35
   4.7 The implementation of the EPP 36

5. Conclusion 37

References 39

Appendices

Appendix 1: (input sample Euclid code) 40
Appendix 2: (output sample Euclid code) 49
Appendix 3: (Euclid Pretty-Printer code) 58
Appendix 4: (Syntax/Semantic Language code) 117

LIST OF FIGURES

Page

Figure

1. The data structure of EPP 7
2. Example of compressing tokens 9
3. Example of pretty-printing expressions 11
4. Example of semantic charts 15
5. A computational model for S/SL 18
6. Example of using S/SL to evaluate a Posfix expression . 21
7. The process of generating an EPP 27
8. Example of using 2 semantic mechanisms:
the positions stack and the marks stack 29

Chapter 1

Introduction

1.1 Background

Whenever a programmer inputs or edits a program on a key-punch or a CRT terminal, he also has to consider how to indent this program to make it look nice and neat. This is really a boring, and time-consuming job. Is there any way that a programmer can just input or edit a program without being concerned about its format? One possible solution is the use of a program called a pretty-printer. A pretty-printer program is designed to parse and reformat the whole text of a source program for a particular language. The advantages of a pretty-printer are that it saves a lot of time editing programs, and it enforces standard indentation conventions.

1.2 Purpose

The purpose of this project is to design a pretty-printer program written in Pascal for Euclid. Euclid is a progamming language designed for writing system programs that could be verifiable by state-of-the-art verification methods. This report is the part of a project whose purpose is to make Euclid available for compilation and execution on the Interdata 8/32

machine under Unix V7 operating system in the Kansas State University's Computer Science Department.

1.3 Approach

There is a pretty-printer program now available written in Pascal for Pascal programs. The overall approach of this project is to modify the existing Pascal Pretty-Printer(PPP) and use the language called Syntax/Semantic Language instead of Pascal to write the parser. In detail, the project was divided into the following 6 phases:

(1). The modification of the Syntax/Semantic Language(S/SL) program to replace the recursive descent parser portion in the existing PPP.

(2). The modification of PPP's lexical scanner to make a new lexical scanner for Euclid.

(3). The modification of PPP's semantic routines and the insertion of new semantic routines to make them compatible with interfaces defined and used in the S/SL program.

(4). The translation of the S/SL program into an S/SL table via an S/SL assembler and the inclusion of this S/SL table or S-code in the Euclid Pretty-Printer(EPP) program.

(5). The inclusion of the S-code interpreter in the EPP.

(6). The implementation of the EPP.

1.4 Organization

This report is organized as follows: Chapter 1 contains explanations of the motivation for designing a pretty-printer, the purpose of this report, and the approach used to accomplish this purpose. Chapter 2 explores the reason why a pretty-printer should be used, the history of pretty-printers, and functions of a pretty-printer. Chapter 3 provides the background of S/SL, why S/SL is used, what S/SL is, features of S/SL, and the implementation of S/SL. Chapter 4 describes in detail the entire process of producing an EPP. Chapter 5 presents the conclusion that summarizes the possible use of S/SL, the results of running EPP, and suggests some future work with the use of S/SL in the pretty-printer area. Appendix 1 is a listing of the input sample Euclid program. Appendix 2 is a listing of the output sample Euclid program. Appendix 3 contains a listing of the EPP program. Appendix 4 consists of a listing of the S/SL program.

chapter 2

Pretty-printing

2.1 The motivation for pretty-printing

When a programmer edits a source program on a key-punch or a CRT terminal, he has to consider some formatting problems such as spacing and indentation to make a program neater and more readable. If he had a formatter program at hand which enables him to do such formatting, the above problems will no longer exist. Therefore, not only can a pretty-printer program save a lot of time in editing, but it also can apply standard indentation conventions to print a new version of the source program in a nice form. Conceivably, designing a pretty-printer program for a language is a worthy effort as it vastly improves the readability and the neatness of programs, though its development is not trivial.

2.2 The history of pretty-printers

There is a Pascal Pretty-Printer program now available on the Interdata 8/32 under Unix V7 at the Kansas State University's Computer Science Department. It was written by Dr. Rodney M. Bates in May, 1981. During his many years of experience in compiler teaching and researching, he felt that keying in a

program while also doing the formatting job simultaneously was a laborious, boring task. For this reason, he was interested in designing a program for reformatting programs. After the PPP program was done, many students received benefits from it.

2.3 Functions of Euclid Pretty-Printer

Euclid Pretty-Printer is a program which serves as a formatter for input source programs to produce a neat, readable version. Its tasks include the following routines:

(1). Insymbol -- This routine is a lexical scanner. It reads tokens from the input source program and classifies them.

(2). Blankline -- It handles the placement of blank lines.

(3). Mark -- It marks some tokens in order to compress later a set of tokens starting from these marked tokens.

(4). Skip -- It skips some unexpected tokens when syntax errors are discovered.

(5). Setup -- It sets up a start position for a token.

(6). Movecmnts -- It changes positions of some comments if necessary.

(7). Indent -- It indents some fixed amount of spaces for some tokens.

(8). Ccompress -- It compresses some tokens if they can be fitted in one line.

(9). Unload -- It unloads and prints a set of tokens which are not considered necessary to be compressed later or their length is more than one line.

The data structure of the Euclid Pretty-Printer is illustrated in Figure 1. It consists of many 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. Therefore, if the length of a token is more than that, we need to create as many nodes as necessary, with their rightlink field linking them together. The downlink field is used to connect between tokens. Gfirstp is a pointer to the first token in the data structure. Gcurp is a pointer to the current token. Gnextp is a pointer to the token which is next after the current token. Gcompp is a pointer to the token from which a set of tokens needs to be compressed later. A mark is a special kind of node whose pointer field always points to the first token of a set of tokens which will be compressed later.

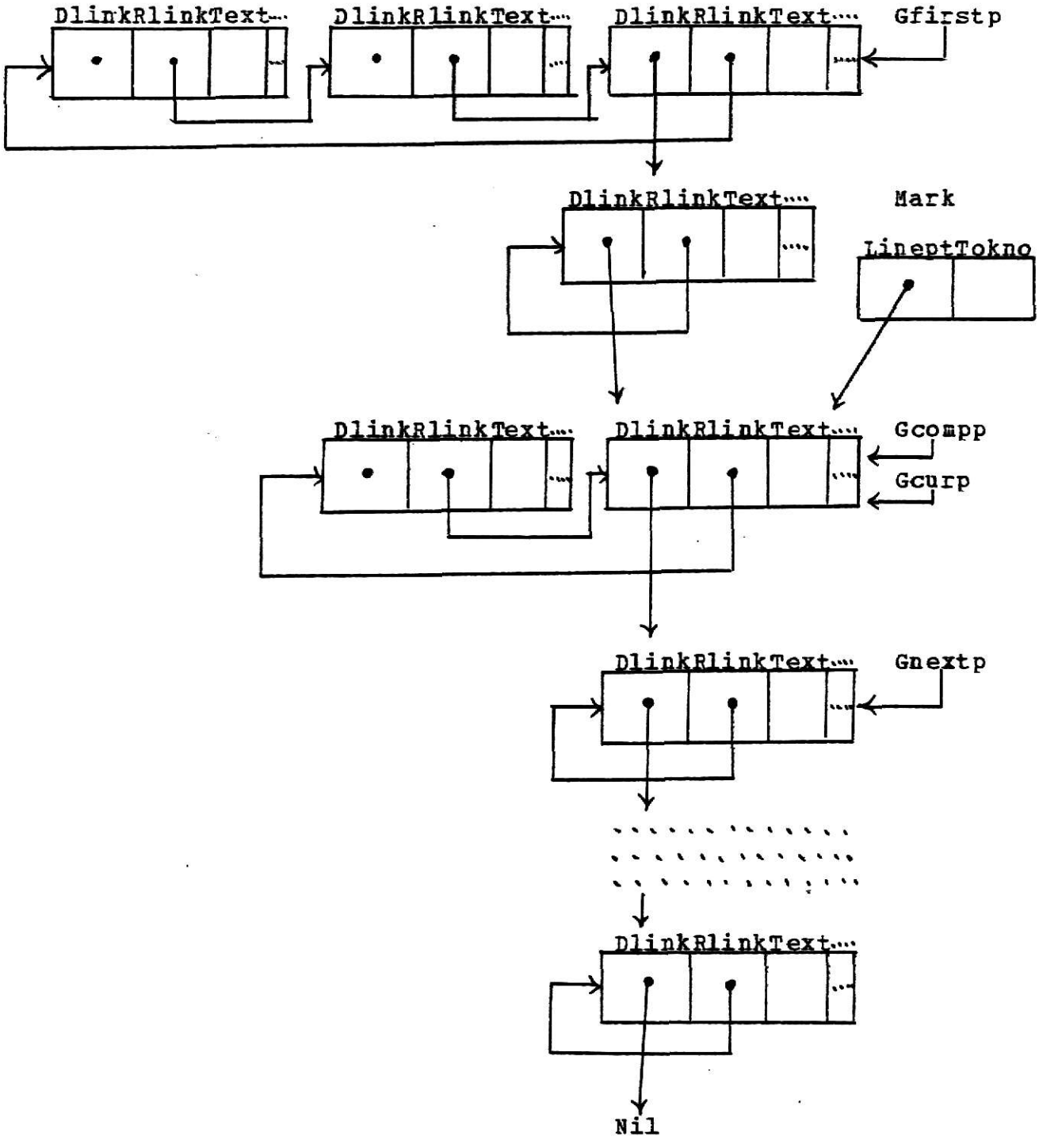


Figure 1. The data structure of EPP

The way Euclid Pretty-Printer compresses tokens is very simple and straightforward. Normally, it first marks a possible starting token from which we might compress a set of tokens later. Then EPP sets up the start position for each token that follows including a possible end token. It does this assuming no syntactic structure will fit in one line. After that, EPP compresses this set of tokens onto a line if they can be fitted in one line; otherwise, they are already arranged into several lines. The example in Figure 2 provides a basic idea of how EPP compresses tokens.

before compression

```
Begin
  x:=x-1
; y:=y+1
; z:=y
End
```

after compression

```
Begin x:=x-1;y:=y+1;z:=y End
```

(b).

before compression

```
Begin
  read(x)
; If A
  Then B
  Else C
  End If
; write(x)
End
```

after compression

```
Begin read(x)
; If A Then B Else C
  End If
; write(x)
End
```

Figure 2. Example of compressing tokens

One of the fascinating things in Euclid Pretty-Printer is the placement of the special character ";". In contradiction to the convention of Pascal, which usually places a semicolon at the end of each line if necessary, EPP places the semicolon at the beginning of each line. The reason for doing this in EPP is to make the format of a text more readable and consistent by placing operators such as a semicolon at the leftmost position of each line followed by an operand. Furthermore, not only can EPP format statements, but it can format basic primitives such as expressions. The formatting rule being enforced on an expression is the same as is used on a semicolon. Examples in Figure 3 illustrate how EPP works neatly on semicolons and expressions.

before pretty-printed

(a).

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

after pretty-printed

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

(b).

```
w := a*(b+2);
y := (u-v/w)+c;
```

```
  w
  := a
     * ( b
         + 2
       )
; y
  := ( u
       - v
         / w
     )
     + c
;
```

Figure 3. Example of pretty-printing expressions

From the first example, we can see very clearly how the indentation of an expression works nicely on the precedence rule of operators, i.e., the rightmost operator in a pretty-printed expression will be executed first at runtime. The second example shows the consistency of parentheses. In addition, both of the examples use semicolons as an separator between statements. As a result, the idea of separating operators from operands used in EPP enhances the readability of a program, and the placement of special characters such as semicolons and parentheses assures the consistency of the context among expressions of a program.

Chapter 3

Syntax/Semantic Language

3.1 Syntax/Semantic Charts

Syntax/Semantic Language(S/SL) is a language which was developed at the University of Toronto as a tool for implementing compilers. Prior to S/SL, Syntax and Semantic Charts had been used by the authors of "An introduction to S/SL: Syntax/Semantic Language", most remarkably for implementing the SP/K subset of PL/1. We will introduce below the concept of Syntax/Semantic Charts and their implementation issues.

Syntax charts were developed in preference to LR(k)-style parsing so that syntax error-handling could be done easily. The charts use separable transition diagrams to define the syntax of a programming language. Each syntax chart describes a finite state machine to recognize a specific fragment of the source language. The charts call one another recursively, forming a push-down automaton to recognize the language. Semantic charts were developed to gain the convenience and efficiency of syntax charts in semantic analysis. The charts describe semantic processing in terms of á set of operations that manipulate a small number of special-purpose data structures. On the other hand, they form a variable-free programming language, which accesses data via a set of operations organized into mechanisms

to manage the associated data structures. These mechanisms in turn can be modelled using abstract data types.

Here, we introduce semantic charts with an example in Figure 4. It is a set of semantic charts to process simple binary operator expressions involving only operators and constants. An assumption is made such that these charts receive from the syntax analysis phase(parser) a stream of tokens which describe an expression in Postfix form. Obviously, we can see from Figure 4 that the semantic charts include the following features: call and return actions; input tokens and input choices; emit(outputs/error signals) actions; semantic operations(simple, parameterized, emit, semantic choice).

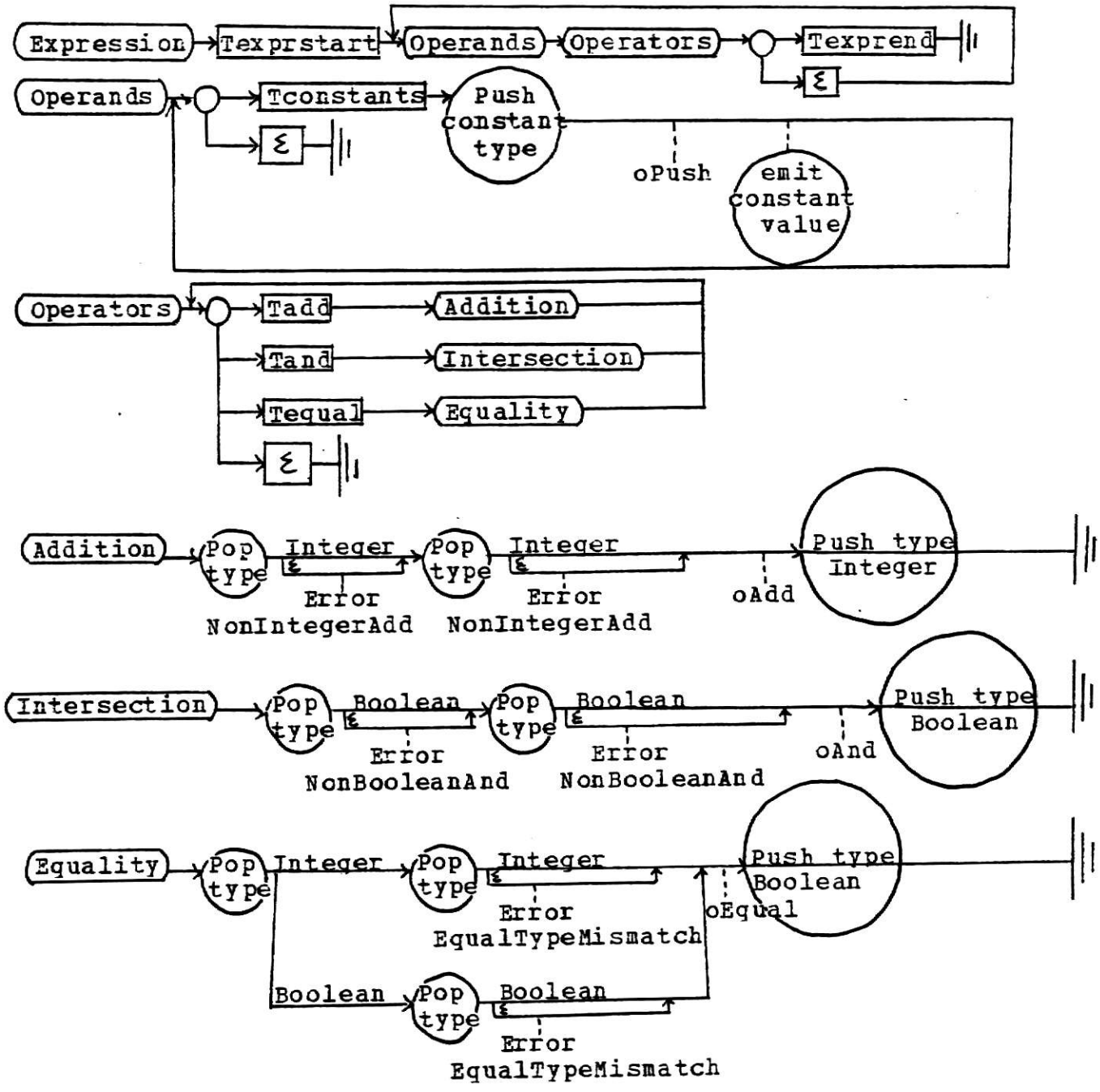


Figure 4. Example of semantic charts

Obviously, overall control in the semantic charts resides in a push-down automaton(table walker) which examines the input stream and uses information in the charts to pass control from mechanism to mechanism by invoking semantic operations. Not only does the technique of semantic charts provide facilities for implementing most of the semantic processing commonly found in compilers(e.g., symbol table management, type checking, branch and fixup management, and pseudo-code generation), but it supplies mechanisms for detecting semantic errors in the source program and for diagnosing and repairing such errors.

There are several ways in which semantic charts can be implemented. The easiest one is to encode the charts as a series of procedure calls to the semantic operations. However, the overhead involved in such an implementation would cause the result to be both large in size and slow in execution. An alternative is to encode the charts into a semantic table by assigning a chart interpreter code to each chart primitive and semantic operation. The charts can in turn be interpreted by a relatively simple table walking algorithm. By choosing an appropriate representation of table operations, the chart can be encoded very compactly.

The SP/K compiler, which was designed to be a high-speed, highly diagnostic compiler, using the scheme of syntax and semantic charts instead of the hand-coded way for its parser and semantic analysis passes, had proved that its compilation rate is significantly better than high-speed, hand-written compilers such

as PL/C and IBM's PL/1 Checkout compiler. One of the reasons for this is that the SP/K compiler handles a limited subset of PL/1.

Although syntax and semantic charts are highly readable and have an efficient implementation, their disadvantage is that they are not machine readable and, like flowcharts, are difficult to maintain. These shortcomings led to the development of S/SL.

3.2 The concept of S/SL

Syntax/Semantic Language(S/SL) is a language developed at the University of Toronto as a tool for constructing compilers. A subset called Syntax Language(SL) has the same recognition power as LR(K) parsers. Complete S/SL includes invocation of semantic operations implemented in another language such as Pascal.

S/SL implies a top-down programming methodology. First, a variable-free algorithm is developed in S/SL. The algorithm invokes operations through semantic mechanisms. A semantic mechanism is an 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 in a different language. On the other hand, the S/SL programmer needn't be directly concerned with the implementation of the operations. All he needs to know in writing his S/SL program is their meaning(specification).

S/SL assumes a computational model that is illustrated in Figure 5. The S/SL program reads 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. Further, an S/SL program consists of a set of possibly recursive rules (subprograms). In order to support this recursion, the implementation uses an implicit stack of return addresses, to allow a called rule to return to the appropriate calling point.

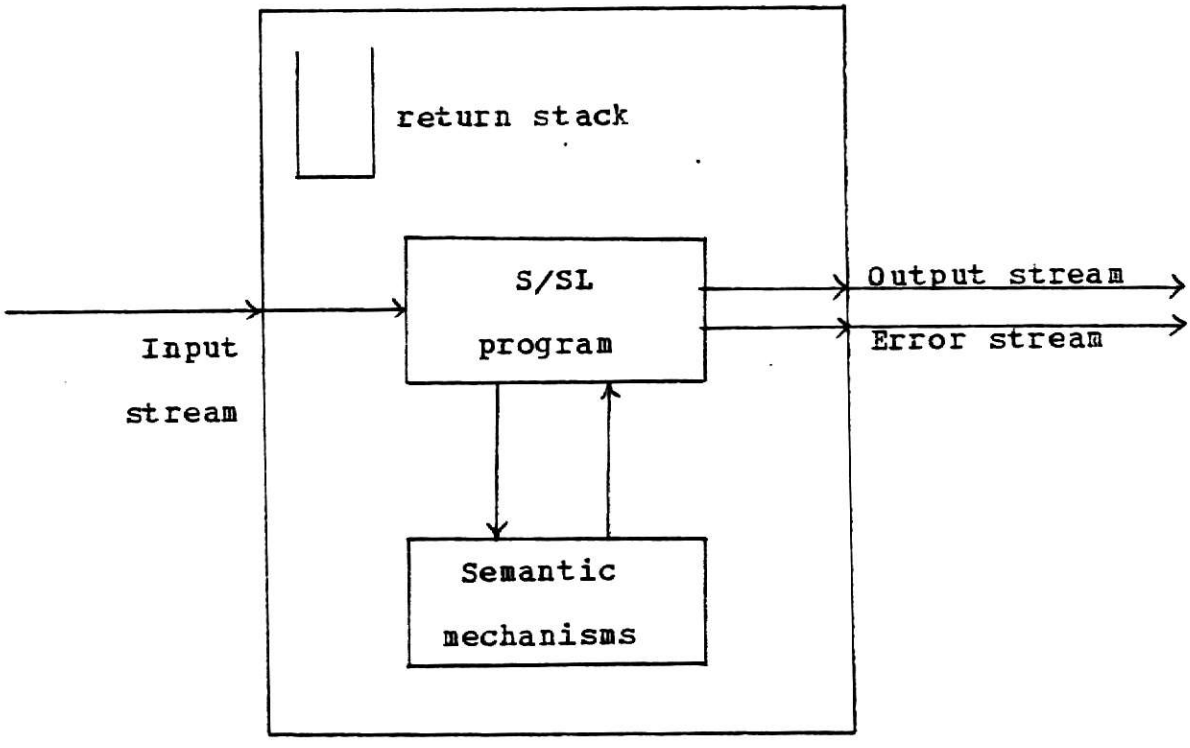


Figure 5. A computational model for S/SL

Besides controlling the input and output streams, the S/SL program can manipulate data via semantic operations which are organized into 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. It is a pure control language without data handling capabilities such as assignment or comparison.

3.3 The features of S/SL

S/SL is a very modest language, including only the following features: sequences, repetitions, and selections of actions (statements); input, matching and output of tokens; output of error signals; subprograms(called rules); and invocations of semantic operations implemented in a base language like Pascal. In addition, S/SL is a programming language which uses the same idea as does Syntax and Semantic Charts but with text notations. It has no variables, nor assignment. The choices in the S/SL program are always deterministic.

Here, we give an example of S/SL use in Figure 6 to show what a S/SL program looks like in the semantic analysis pass of a compiler. An assumption has been made such that the input to our semantic analysis pass has its expressions in syntactically correct Postfix. To simplify our example, we consider here only

3 operators: addition, equality and intersection(and). As in Pascal, we assume that only numbers can be added, only booleans can be intersected, numbers can be compared to numbers only and booleans to booleans only. Also, we assume that all numbers are integers.

```
PosfixExpression:
  {
   @Primaries @Operators
   [=
     |exprend:
              >
     |*:
   ]
  };

Primaries:
  {[=
     |constant: TypePushConstant
     .....other primaries.......
     |*:
        >
   ]};

Operators:
  {[=
     |add:
          @CheckInteger @CheckInteger TypePush(Int)
     |and:
          @CheckBoolean @CheckBoolean TypePush(Bool)
     |equal:
           @CheckEquality TypePush(Bool)
     |*:
        >
   ]};

CheckInteger:
  [= TypeChoose
     |Int:
     |*: #IntegerRequired
  ]
  TypePop;

CheckEquality:
  [= TypeChoose
     |Int:
          TypePop @CheckInteger
     |Bool:
          TypePop @CheckBoolean
  ];

CheckBoolean:
  [= TypeChoose
     |Bool:
     |*: #BooleanRequired
  ]
  TypePop;
```

Figure 6. Example of using S/SL to evaluate a Posfix expression

As each primary is accepted, its type is pushed onto the type stack. As each operator is accepted, the types of its operands are checked and their types on the type stack are popped and replaced by the operator's result type. The manipulation of the type stack is done entirely by semantic operations.

3.4 Why S/SL is used in this project?

S/SL is a synthesis of several programming concepts. The following related notations have been directly influential:

(1). BNF and regular expressions.

(2). Syntax and Semantic Charts.

(3). Recursive descent compilers.

(4). Separable transition diagrams.

(5). Table-driven coding.

(6). Data encapsulation techniques.

Although S/SL absorbs such a large portion of the power of these notations, it itself remains very simple. Its simplicity results in readability, a great deal of flexibility and an easy, efficient implementation.

S/SL is a very small language such that it contains only eight actions, plus calls to semantic operations. The concept of

a pure control language and abstract data types in S/SL implies that S/SL is not a general-purpose language and therefore only suited for a small set of programs.

The above concept led us to replace the recursive descent parser written in Pascal in the Pascal Pretty-Printer with a parser written in S/SL for the Euclid Pretty-Printer.

3.5 The implementation of Syntax/Semantic Language

S/SL has been used to construct compilers for SP/K(a PL/1 subset), PT(a Pascal subset) and Toronto Euclid. It has been used to implement scanners, parsers, semantic analyzers and code generators.

The implementation of S/SL in this project is essentially the same as the implementation of Syntax and Semantic Charts via a table and a table walker. First, the S/SL program is translated into an S/SL table(called S/SL machine, or S-code) via an S/SL assembler. This table can be stored in a Pascal array. Hence, it is accessed by a Pascal program called a table walker(or S-code interpreter). The S-code interpreter walks through the table executing instructions and thus carrying out the actions of the S/SL programs.

When implementing the S/SL program, the S/SL assembler uses Hartmann's[7] error recovery techniques. Indeed, the original implementation of S/SL, developed at the University of Toronto,

did not do this job. Later, Grace Evans [6] modified the assembler, the S-code language and the S-code interpreter to accomplish this job. This accomplishment insures that we do not have to worry about error diagnosis and repair problems in the S/SL program.

Chapter 4

Euclid Pretty-Printer

4.1 The process overview

The entire process of generating an Euclid Pretty-Printer(EPP) using Pascal is intricate but interesting work. First, we had to understand exactly what Pascal Pretty-Printer(PPP) is doing and what S/SL is. Also, we did not know what and how many interfaces need to be defined in the S/SL program without first understanding how semantic mechanisms are related to one another. After the replacement of the recursive descent parser program in PPP with the S/SL program, modifications and insertions were necessary for semantic routines in PPP in order to implement semantic mechanisms defined in the S/SL program. Furthermore, we needed to modify the lexical scanner in PPP for Euclid programs due to the difference in reserved words and special characters between Pascal and Euclid.

Now, it was time for the implementation after we have done the above programs. First, we translated the S/SL program into a S/SL table using the S/SL assembler. This S/SL table could in turn be accessed by a Pascal program called a table walker or S-code interpreter to execute the entire S/SL program. The final step was to compile and run the EPP written in Pascal on the Interdata 8/32 machine under Unix V7. Figure 7 illustrates the

whole process of producing an EPP. The paragraphs below will describe this process in detail.

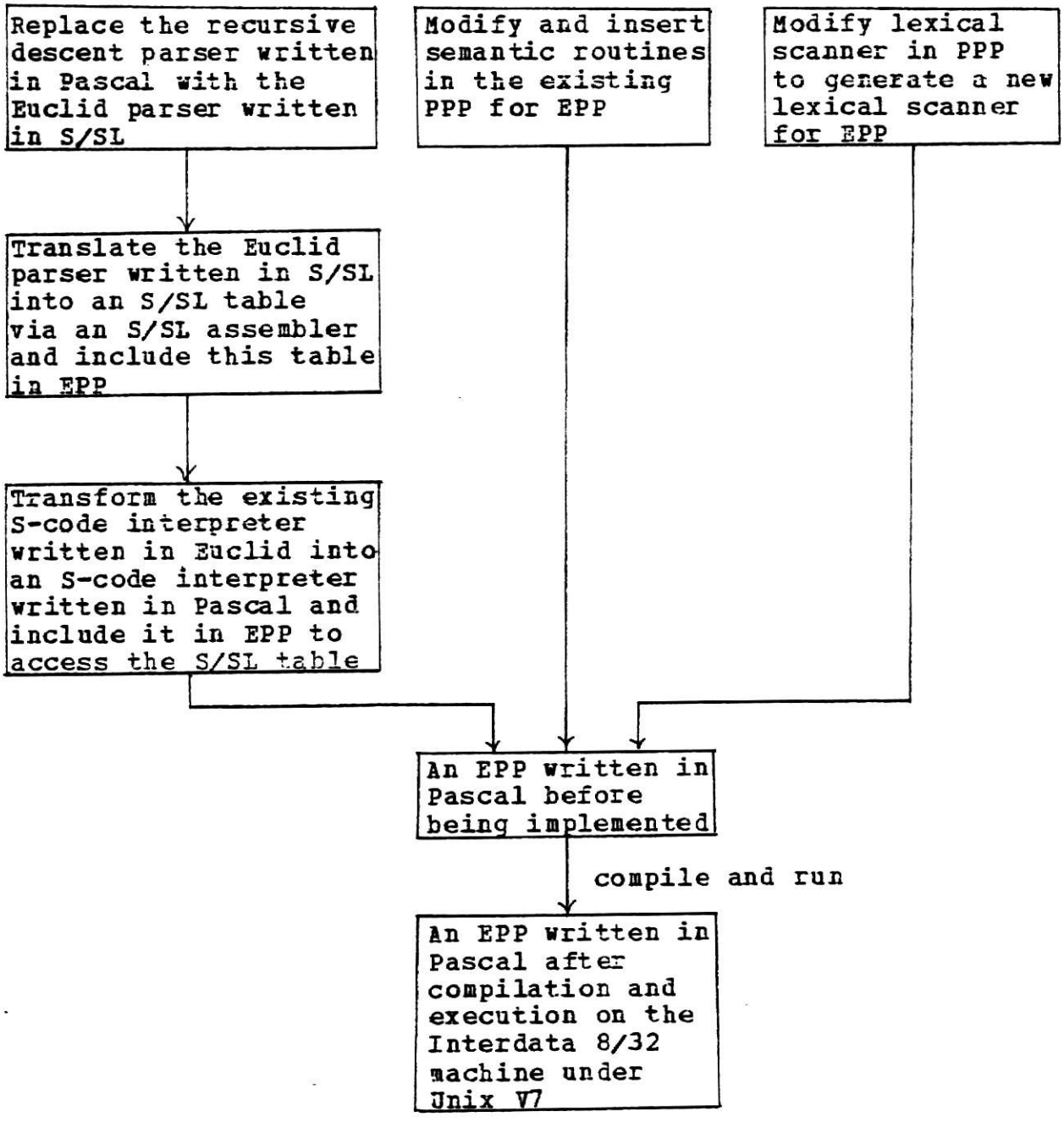


Figure 7. The process of generating an EPP

4.2 The use of S/SL

The main reason we used the Syntax/Semantic Language(S/SL) program to serve as a recursive descent parser for EPP was that a parser for an Euclid compiler was written in S/SL. With the existing Euclid parser at hand, we just added some semantic mechanisms into it.

The problems arose when we started writing the S/SL program in place of PPP's recursive descent routines. PPP routines had positions and marks as local variables, but S/SL has no variables. How do we simulate local variables of PPP's routines when writing S/SL rules? How do we pass parameters to S/SL called rules? Because S/SL is a variable-free language, some semantic mechanisms must be included in the original S/SL program to help handle these kinds of problems. Thus, stacks of positions and marks were added as a part of semantic mechanisms in the S/SL program. Values from stacks were used as parameters to existing PPP's semantic routines such as Ccompress and Indent.

We give an example here to show how stacks of positions and marks implement efficiently to handle the problems of local variables and parameter passing. To simplify problems, the example in Figure 8 uses only a syntactic structure of the If statement in Euclid. We will discover below how a syntactic structure is formatted at first and how it is formatted after being compressed.

(a). The original, unformatted If statement

```
                    ;
If A
   Then B
        Else C
End If              ;
```

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

```
; If A
  Then B
  Else C
  End If
;
```

(c). Four possible formats of If statement after compression

(1).

```
; If A Then B Else C End IF
;
```

(2).

```
; If A Then B Else C
  End IF
;
```

(3).

```
; IF A Then B
  Else C
  End IF
;
```

(4).

```
; IF A
  Then B
  Else C
  End If
;
```

Figure 8. Example of using 2 semantic mechanisms: the positions stack and the marks stack

In the example, we first make an assumption that we are already in a rule called Statements. Thus, we encounter an input choice. Prior to setting up the position for the token 'if', we need to indent 'if' to make it right after a semicolon. The simple way to do this is to fetch the top value of the positions stack, which is exactly the position value for the semicolon passed from the caller rule, and use it as a parameter to a PPP routine called Indent. The indented position value returned from this routine is in turn pushed onto the top of the positions stack. After that, we set up the position for 'if' by fetching the top element of the positions stack and using it as a parameter to a PPP routine called Setup. These processes just explain how the positions 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, we will invoke a rule called IfStatement to handle the body of the If statement. We ought to consider some problems before doing this. First, we will have to compress later the token 'if' 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, we usually place an expression right after the operator 'if' if they can be fitted in one line. This implies 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 marks 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 to see if the whole If statement(including 'if') can be fitted in one line, after the second mark has been popped from the marks stack when the routine IfStatement was returned. The second current token mark being pushed onto the marks stack is used as a parameter to the routine IfStatement, and then serves as a starting point from which some token might be compressed later. In figure 8.(c), we can see 4 different possible formats resulting from the above process. Apparently, the process described above tells us how the marks 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 the problems such as local variables and parameter passing, we need to push a position value onto the top of the positions stack to serve as an additional parameter besides the other, which is the top value of the marks 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 8, it pushes the position value for 'if' onto the positions stack and the current token mark, whose pointer field points to 'if', onto the marks stack, and passes both of them as parameters to a rule called IfStatement. Later, within this rule, we will make use of these two values 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

marks stack and the positions stack are used to pass both the position value and the mark value to a called rule, and then this rule uses both values as its local variables to format its substructures. One thing we have to be aware of is whenever we return to a caller from a rule, the top value of both the positions stack and the marks stack is always popped, because these two values are only meaningful to a called rule. Figure 8.(a) shows an original, unformatted program. Figure 8.(b) shows how this program is formatted at first. Figure 8.(c) shows 4 possible formats after this program is compressed.

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 Euclid parser, written in S/SL was able to parse and reformat the input Euclid program in a straighfoward manner.

Originally, it was difficult for us to understand how to format an expression or a statement using these semantic operations. But once we have drawn a diagram to see how a fragment of a syntactic structure ought to be formatted, it turned out to be much easier to write the S/SL part of this project.

4.3 Modifications and insertions of PPP's semantic routines

The semantic mechanisms defined and used in the S/SL program must be implemented in Pascal. First, we had to define two stacks and their top elements for the positions stack and the marks stack mechanisms defined in the S/SL program. In addition, some semantic routines must be added in order to implement the semantic mechanisms defined and used in the S/SL program. Also, slight modifications on some PPP routines were necessary to cope with insertions of the above semantic routines. We will briefly explain below the functions of these routines.

The routine PushConstPos pushes integer constant 1 onto the positions stack. PushPos accepts a parameter which is a position value and pushes it onto the positions stack. PushMark accepts a parameter which is a mark record and pushes it onto the marks stack. PopPos pops the top value from the positions stack. PopMark pops the top value from the marks stack. These 5 semantic routines are added to the existing PPP. The routine Mark originally only accepts a blank mark record and returns a mark record with its pointer field pointing to Gcurp↑ and its tokno value set to Gcurp tokno. In order to implement two different interfaces defined in the S/SL program, namely, oPushCurrentTokenMark and oPushNextTokenMark, we had to modify the routine Mark to accept the second parameter which either points to Gcurp↑ or Gnextp↑ depending on what the second parameter is.

On the other hand, we did not need to change the contents of some semantic routines such as Unload, Setup, Skip, Ccompress,

etc, because as far as the pretty-printer is concerned, contents of these semantic routines would not be affected by the S/SL program.

4.4 The modification of PPP's lexical scanner

Here, we needed to modify the existing lexical scanner in PPP into a lexical scanner for Euclid. First, we had to understand that Euclid has different reserved words from those in Pascal. We had to replace those reserved words used in Pascal with those used in Euclid. Also, there are some special characters in Euclid which Pascal does not have and viceversa. For example, "=>" (label) or "=>" (imply), which are tokens in Euclid which are not seen in Pascal, should be added. Also, since Euclid only allows characters "{" and "}" to serve as comment delimiters, symbles "(*", "*)", and """ which delimit comments in Pascal needed to be deleted. Further, Euclid uses the dollar sign($) to represent character constants and special character values. This was handled by another routine called ProcessDollarSign.

Indeed, writing the modified lexical scanner was simplified by the existence of a program called Xlate. This program was written by David Roesener[8] . It translates April Euclid(a subset of Euclid) into KSU Interdata 8/32 Pascal. Since Xlate contains Euclid lexical scanner, we used it as a guide for creating the EPP's lexical scanner.

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

Before the S/SL program was included in the EPP, the original recursive descent parser portion in PPP had to be deleted. Because the S/SL program was not Pascal-coded, it was necessary to translate it into S-code form 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.

We call the translated S/SL program an S/SL table, since the sequence of numbers in this table was stored in a Pascal array. This table was then included in the declaration part of the EPP. Later, a Pascal program called a table walker or an S-code interpreter will access this table and execute its instructions.

4.6 The inclusion of the S-code interpreter

Here, we used the existing S-code interpreter written in Euclid and changed it into a S-code interpreter written in Pascal. In addition, we needed to add some semantic routine calls to this S-code interpreter which are unique to the EPP S/SL program. Also, due to the insertions of two semantic mechanisms in the S/SL program, i.e., the positions stack and the marks

stack, we needed to modify the routine EmitError to emit error messages such as "EPP position stack overflow at line 1270". This modification in turn introduced two routines which must be added to EPP. Namely, DisplayString that emits a string of characters and DisplayInteger that emits an integer value, which is indeed the line number of the S/SL program.

We call this S-code interpreter a table walker. It had to be included in the Euclid Pretty-Printer program as a subroutine. We invoke this table walker in the main program of the Euclid Pretty-Printer and it will walk through the S-code table and execute its instructions, and thus carry out the actions of the S/SL program.

4.7 The implementation of the EPP

Actually, the S/SL assembler and the S-code interpreter used in this project automatically produce Hartmann style recovery from syntax errors. We thus avoided the error recovery problems in the Euclid parser. Eventually, when we combined steps 3, 4, 5, 6, the result was a Euclid Pretty-Printer program written in Pascal. This program was compiled and run as usual, giving an Euclid Pretty-Printer.

Chapter 5

Conclusion

In this project, we have used an S/SL program to serve as a recursive descent parser portion of the Euclid Pretty-Printer written in Pascal. This turns out to be a very convenient scheme in which we just define the meaning of semantic mechanisms without directly being concerned with implementation details.

Although S/SL evolved as a tool for constructing compilers, it appears to be useful for a much larger class of problems. The concept of well-defined abstract data types involved in an S/SL program is one of the most important tools available for structuring programs and is used constantly in software engineering.

Therefore, we expect to experiment with S/SL as a software specification language in the future. As such an S/SL program will serve as the top level, an executable design for a system, which is to be implemented later by programming its semantic mechanisms. For large systems, these implementations may in turn be designed using S/SL, the result being several levels of S/SL implementing increasing levels of details of the system.

The S/SL program included in EPP consists of 3102 lines. EPP itself contains 2482 lines. We have used one sample Euclid program as input to run EPP. The input program has 264 lines,

and the output program generated has 307 lines. There are 807 tokens in the input program which were parsed by EPP, not including comment tokens. EPP took 52k bytes of memory and 12 seconds to run this sample program.

From the nature of the data structure of EPP, we know that the size is not a function of input program lengths, but a function of syntactic depths. This means that the size will remain constant in proportion to the growth of input programs. On the other hand, the size will increase in proportion to the increase of nested depths of input programs.

There are two problems which remain unresolved in EPP. One is that consuming a token when we do Setup causes the error recovery to fail. This problem was discovered when EPP was run on a sample program which is syntactically incorrect. The reason the error recovery failed is simply that we did change the S-code interpreter not to consume a token when an input match or choice is encountered. The other problem is that the original Euclid parser from Toronto has errors such as occurences of semicolons in some unexpected places. These two problems will be solved in the future.

As far as a pretty-printer is concerned, we would suggest two things that can be completed. One is to write a new PPP using S/SL. The other is to rewrite the Pascal portion of EPP in Euclid.

References

1. Rodney M. Bates; A Pascal Pretty-Printer with a different purpose; SIGPLAN notices, Vol.16, No.3, March 1981, pp. 10..17.
2. Rodney M. Bates; Pascal PPP(a Pascal program); May 23, 1981, Kansas State University, Manhattan, Kansas.
3. J. R. Cordy; Euclid Parser Tables V1.24; Feb. 27, 1979.
4. J. R. Cordy, R. C. Holt, and D. B. Wortman; An Introduction to S/SL: Syntax/Semantic Language; ACM Transactions on Programming Language and Systems, Vol.4, No.2, April 1982, pp. 149-178.
5. J. R. Cordy, R. C. Holt, and D. B. Wortman; Semantic Charts: A Diagrammatic Approach to semantic processing; ACM SIGPLAN notices, Vol.14, No.8, August 1979, pp. 39-49.
6. Grace Evans; A parser modification of the Euclid compiler: automatic generation of syntax error recovery; The master report, 1982, Kansas State University, Manhattan, Kansas.
7. Alfred C. Hartmann; A concurrent Pascal compiler for Minicomputers; The doctor thesis, Sept. 22, 1975, California Institute of Technology, Pasadena, California.
8. David P. Roesener; April Euclid to Pascal Translator; The Master report, 1980, Kansas State University, Manhattan, Kansas.

Appendix 1

input sample Euclid code

```
      1 type WORLD = module                                                    *      *
      2
      3 var InteractiveLineNoResolver:                                         *
      4     module                                                             1
      5
      6         imports (var IO);                                              *      5
      7
      8         {   Project Euclid
c     9
c    10             Module: Interactive Line Number Resolver V1.09
c    11             Author: David R. Crowe
c    12             Reader: J.R. Cordy
c    13             Date:   14 January 1980
c    14
c    15         This module transcribes input from the terminal to the terminal
c    16         output device, suffixing each stream line number with its
c    17         resolved input line number & file name.  A string of digits is
c    18         considered to represent a stream line number if it appears at the
c    19         beginning of a line or is immediately preceded by a colon.  Thus,
c    20         for example, the input,
c    21                 Euclid:483: Symbol _6270ul undefined
c    22         might be transcribed as
c    23                 Euclid:483 [line 481 of INCLFILE]: Symbol _6270ul undefined  .}
     24
     25
     26
     27         { Largest SignedInt value (i.e. SignedInt.last). }
     28
     29         pervasive const lastSignedInt := 32767;                         *      *
     30
     31         { Source line counter modulus. }
     32
     33         pervasive const lineNoModulus := 30000;                        29      *
     34
     35         { Maximum number of line no. directory entries that will be processed.
c    36         This number does not include the "0th" entry (0,0,""), which is not      *
c    37         present in the line no. directory file.  This limit also in errlist.e }
     38
     39         pervasive const maxLineNoDirectoryEntries := 127;  { Allows 63 includes }      33      *
     40
     41         { Input/output channel number for line no. directory file. }
     42
     43         pervasive const lineNoDirectoryFile := 1;                      39      *
     44
     45         { Predefined character constants. }
     46
     47         pervasive const endOfFile := $$E;                              43      *      *
     48         pervasive const newLine := $$N;                                47      *      *
     49
```

```
    50          var LineNoResolver:                                                 *
    51              module                                                          1
    52
    53                  imports (lineNoModulus, IOMode, maxFileNameLength, var IO);         6    33     *     *     5
    54
    55                  exports (ResolveLineNo, lineNoDirectory, lineNoDirectoryIndex,           *     *     *     *
    56                      LineNoDirectoryEntry  ( with (fileName, fileNameLength) ),           *
    57                      LineNoDirectoryTable  ( with (IndexType) ), inputLineNo);       *    *
    58
    59                  { This module handles the resolution of line numbers appearing
c   60                  in the error streams (called "stream line numbers") to their        *     *     *
c   61                  corresponding source input line numbers & file names.  Line
c   62                  numbers are resolved using a table called the line number
c   63                  directory.  This table is read from a file created by the
c   64                  Screener.  Its values are based upon the placement of include
c   65                  clauses in the source program.  The algorithm used to resolve
c   66                  line numbers, using the directory, is described in
c   67                  Project Euclid Working Paper 42. }
    68
    69                  { Type declarations for a line no. directory entry & for the
c   70                  line no. directory table. }
    71
    72                  type LineNoDirectoryEntry =                                 56
    73                      record
    74                          var streamLine: 0..lineNoModulus-1;                  *    53
    75                          var inputLine:  0..lineNoModulus-1;                  *    53
    76                          var fileName: array 1..maxFileNameLength of Char;         *    53     5
    77                          var fileNameLength: 0..maxFileNameLength;            *    53
    78                      end LineNoDirectoryEntry;                               72
    79
    80                  type LineNoDirectoryTable =                                 57
    81                      array 0..maxLineNoDirectoryEntries of LineNoDirectoryEntry;          39    78
    82
    83                  { Variable declaration for line no. directory. }
    84
    85                  var lineNoDirectory: LineNoDirectoryTable;                  55    80
    86
    87                  { Number of entries read into line no. directory. }
    88
    89                  var noLineNoDirectoryEntries: 0..maxLineNoDirectoryEntries;          *    39
    90
    91                  { Values returned by ResolveLineNo procedure. }
    92
    93                  var inputLineNo: 0..lineNoModulus-1;                        57    53
    94                  var lineNoDirectoryIndex: 0..maxLineNoDirectoryEntries;     55    39
    95
```

THE FOLLOWING PAGE IS CUT OFF

THIS IS AS RECEIVED FROM THE CUSTOMER

```
    96                  procedure ResolveLineNo (streamLineNo: 0..lineNoModulus-1) =      55      *     53
    97
    98                      imports (noLineNoDirectoryEntries, lineNoDirectory,      6     89     85
    99                          var inputLineNo, var lineNoDirectoryIndex);         93     94
   100
   101                      pre (lineNoDirectory has been initialized);              *
   102
   103                      post (lineNoDirectory(lineNoDirectoryIndex).streamLine <= streamLineNo and       *    98    99    74    9
   104                          (lineNoDirectoryIndex < noLineNoDirectoryEntries ->      103    98
   105                          lineNoDirectory(lineNoDirectoryIndex+1).streamLine > streamLineNo) and     103   104   103    96
   106                          inputLineNo = streamLineNo -                         99   105
   107                          lineNoDirectory(lineNoDirectoryIndex).streamLine +       105   105   105
   108                          lineNoDirectory(lineNoDirectoryIndex).inputLine);        105   107    75
   109
   110                      { This routine resolves a stream line no. to its
c  111                      corresponding input line no. & file name.  (The file name
c  112                      will be "" for the outer source file.)  The input line
c  113                      no. will be returned in lineNoDirectory(lineNoDirectoryIndex).fileName;
c  114                      its length will be returned in
c  115                      lineNoDirectory(lineNoDirectoryIndex).fileNameLength.
c  116
c  117                      This routine will return possible incorrect results if
c  118                      either of the following conditions holds:
c  119                       1. stream line nos. for the program overflow lineNoModulus,
c  120                          & consequently wrap around;
c  121                       2. the no. of entries in the line no. directory file
c  122                          exceeds maxLineNoDirectoryEntries.  }
   123
   124                      begin
   125                          lineNoDirectoryIndex := noLineNoDirectoryEntries;       108   104
   126                          loop                                                     *
   127                              exit when (lineNoDirectory(lineNoDirectoryIndex).      *     *   108   108
   128                                    streamLine <= streamLineNo);                107   105
   129                              lineNoDirectoryIndex := lineNoDirectoryIndex - 1;   108   108
   130                          end loop;                                              126
   131
   132                          inputLineNo := streamLineNo -                          106   105
   133                                lineNoDirectory(lineNoDirectoryIndex).streamLine +       108   108   107
   134                                lineNoDirectory(lineNoDirectoryIndex).inputLine;         108   108   108
   135
   136                      end ResolveLineNo;                                          96
   137
```

```
138             initially {for LineNoResolver}
139
140                 imports (IOMode, var IO, var lineNoDirectory,
141                     var noLineNoDirectoryEntries);
142
143                 pre (maxLineNoDirectoryEntries > 0);
144
145                 post  (lineNoDirectory(0..noLineNoDirectoryEntries)
c   146                     is initialized);
147
148                 begin
149                     var k: 1 .. maxLineNoDirectoryEntries + 1;
150                     var inputInt: SignedInt;
151                     var i: 1..maxFileNameLength;
152
153                     (Set 0th directory entry to (0,0,'').)
154                     lineNoDirectory(0).streamLine := 0;
155                     lineNoDirectory(0).inputLine := 0;
156                     lineNoDirectory(0).fileNameLength := 0;
157
158                     IO.Open(lineNoDirectoryFile, inFile);
159                     k := 1;
160                     loop
161                         IO.ReadInt(lineNoDirectoryFile, inputInt);
162                         exit when (IO.EndFile(lineNoDirectoryFile));
163                         lineNoDirectory(k).streamLine := inputInt;
164
165                         IO.ReadInt(lineNoDirectoryFile, inputInt);
166                         lineNoDirectory(k).inputLine := inputInt;
167
168                         IO.ReadInt(lineNoDirectoryFile, inputInt);
169                         lineNoDirectory(k).fileNameLength := inputInt;
170                         if inputInt > 0  then
171                             i := 1;
172                             loop
173                                 IO.ReadChar(lineNoDirectoryFile,
174                                     lineNoDirectory(k).fileName(i));
175                                 exit when i = inputInt;
176                                 i := i + 1;
177                             end loop;
178                         end if;
179
180                         assert (not IO.EndFile(lineNoDirectoryFile));
181
182                         k := k + 1;
183                         exit when (k > maxLineNoDirectoryEntries);
184                     end loop;
185
186                     noLineNoDirectoryEntries := k - 1;
187                     IO.Close(lineNoDirectoryFile);
188                 end; (initially)
189
190             end module; (LineNoResolver)
191
```

*									
6	53	S	108						
104									
101	39								
103									
*	143								
*	*								
*	53								
140	107								
154	108								
155	77								
S	*	43	*						
149									
126									
S	*	158	150						
	127	127	S	*	161				
156	159	154	161						
S	161	162	163						
163	163	155	165						
S	165	165	166						
	166	166	156	168					
169									
151									
160									
S	*	168							
	169	169	76	171					
162	162	174	170						
175	*								
172									
	*	S	162	173					
174	*								
175	175	182	143						
177									
141	183								
S	*	180							
1									

```
    initially                                                                   13d
    begin
        imports (var IO, var LineNoResolver);                                    6      S     50
        pre (lineNoModulus <= (lastSignedInt) div 10 * 10);                    143     53     29
        begin
            var inputChar: Char;     { Current input char. }                     *      S
            var accumulatedNo: 0 .. lastSignedInt; { Value of digits read       *    196
                thus far in stream line no. }
            var inStreamLineNoContext: Boolean; { Indicates whether next        *      S
                char might be part of stream line no. }
            var k: 1 .. maxFileNameLength;                                     186     53
            accumulatedNo := 0;                                                200
            inStreamLineNoContext := true;                                     202      S
            loop                                                               184
                IO.GetChar (inputChar);                                          S      *    199
                exit when inputChar = endOfFile;                               183    183    210     47
                if inStreamLineNoContext then                                  207
                    if inputChar >= $0 and inputChar <= $9 then                211      *
                        if accumulatedNo <= lastSignedInt div 10 - 1 then      206    196
                            accumulatedNo := accumulatedNo * 10 +              206    206
                                (Char.Ord(inputChar) - Char.Ord($0));                S      S    214      S      S
                        else
                            { Number is too big.  Forget about resolving it.)
                            accumulatedNo := 0;                                206
                            inStreamLineNoContext := false;                    213      S
                        end if;
                    else
                        if accumulatedNo not = 0 then                          220
                            LineNoResolver.ResolveLineNo (accumulatedNo);      194     96    224
                            IO.PutString (' [');                                 S      *
                            { Output resolved line no. & file name. }
                IO.PutString('line ');                                           S    226
                IO.PutInt(LineNoResolver.inputLineNo);                           S      *    194    106
                if LineNoResolver.lineNoDirectory(LineNoResolver.              230    174      *
                    lineNoDirectoryIndex).fileNameLength > 0  then             108    169
                    k := 1;                                                    204
                    IO.PutString(' of ');                                        S    229
                    loop                                                       209
                        IO.PutChar(LineNoResolver.lineNoDirectory                S      *    231    231
                            (LineNoResolver.lineNoDirectoryIndex).fileName(k));          236    232    174    233
                        exit when (k >= LineNoResolver.lineNoDirectory         211    211    237    237    236
                            (LineNoResolver.lineNoDirectoryIndex).             237    237
                            fileNameLength);                                   232
                        k := k + 1;                                            238      *
                    end loop;                                                  235
                end if;
                            IO.PutChar ($1);                                     S    236
```

```
246                                 accumulatedNo := 0;                    225
247                             end if;
248                             inStreamLineNoContext := false;            221     S
249                         end if;
250                     end if;
251
252                     if inputChar = $: or inputChar = newLine then      217     *    48
253                         inStreamLineNoContext := true;                 248     S
254                     end if;
255
256                     IO.PutChar (inputChar);                              S   245   252
257                 end loop;                                              242
258
259             end; (initially)
260
261     end module; (InteractiveLineNoResolver)                            190
262
263 initially begin end;                                                  192
264 end module; (WORLD)                                                   261
```

cross reference * is def = is asg

-A-
accumulatedNo 200* 206= 215 216= 216 220= 224* 225* 246=
assert 180*

-B-
Boolean 202

-C-
Char 76 199 217* 217*
Close 187*

-E-
E 47*
EndFile 162 180*
endOfFile 47= 211
exit 127 162* 175* 183* 211* 238*
exports 55*

-F-
false 221* 248*
fileName 76* 174* 237*
fileNameLength 77* 156= 169= 232* 240*

-G-
GetChar 210*

-I-
i 151* 171= 174* 175* 176= 176*
imports 6 53 98 140 194*
inFile 158*
initially 138 192* 263*
inputChar 199* 210* 211* 214* 214 217* 252* 252 256*
inputInt 150* 161* 163* 165* 166* 168* 169* 170* 175
inputLine 75* 108* 134 155= 166=
inputLineNo 57* 93* 99* 106* 132= 230*
inStreamLineNoContext 202* 207= 213* 221= 248= 253=
interactiveLineNoResol- 3
ver
IO 6* 53* 140* 158* 161* 162* 165* 168* 173* 180* 187* 194* 210* 226* 229* 230*
234* 236* 245* 256*
IOMode 53* 140

-K-
k 149* 159= 163* 166* 169* 174* 182= 182* 183* 186* 204* 233= 237* 238* 241= 241*

-L-
lastSignedInt 29= 196* 200 215
line 60*
lineNoDirectory 55* 85* 98* 103* 105* 107 108* 127 133 134 140* 154* 155* 156* 163* 166*
169* 174* 231* 236* 238
LineNoDirectoryEntry 56* 72* 78* 81
lineNoDirectoryFile 43= 158* 161* 162* 165* 168* 173* 180* 187*
lineNoDirectoryIndex 55* 94* 99* 103* 104* 105* 107* 108* 125= 127 129= 129 133 134 232* 237*

```
                          239*
 LineNoDirectoryTable      57*   80*   85
 lineNoModulus             33=   53*   74    75    93    96   196*
 LineNoResolver            50*  194*  225   230*  231*  231*  236*  237*  238   239*
 loop                     126   130   160*  172*  177*  184*  209*  235*  242*  257*

-M-
 maxFileNameLength         53*   76    77   151   204
 maxLineNoDirectoryEntr-   39=   81    89    94   143*  149   183*
   ies
 module                     1     4    51   190*  261*  264

-N-
 N                         48*
 newLine                   48=  252
 noLineNoDirectoryEntri-   89*   98*  104*  125   141*  186=
   es
 numbers                   60*

-O-
 Open                     158*
 Ord                      217*  217*

-P-
 pervasive                 29*   33*   39*   43*   47*   48*
 post                     103*  145*
 pre                      101*  143*  196*
 PutChar                  236*  245*  256*
 PutInt                   230*
 PutString                226*  229*  234*

-R-
 ReadChar                 173*
 ReadInt                  161*  165*  168*
 ResolveLineNo             55*   96*  136   225

-S-
 SignedInt                150
 stream                    60*
 streamLine                74*  103*  105*  107*  128   133   154=  163=
 streamLineNo              96*  103   105*  106   128   132

-T-
 th                        36*
 true                     207*  253

-W-
 when                     127   162*  175*  183*  211*  238*
 WORLD                      1*

end xref   63 identifiers   297 total references
  99 collisions.
```

Appendix 2

output sample Euclid code

```
      1 type WORLD                                                                *
      2   = module                                                                *
      3       var InteractiveLineNoResolver                                       *
      4         : module                                                          2
      5             imports ( var IO )                                            *     5
      6         {   Project Euclid
c     7
c     8             Module: Interactive Line Number Resolver V1.09
c     9             Author: David R. Crowe
c    10             Reader: J.R. Cordy
c    11             Date:   14 January 1980
c    12
c    13         This module transcribes input from the terminal to the terminal
c    14         output device, suffixing each stream line number with its
c    15         resolved input line number & file name.  A string of digits is
c    16         considered to represent a stream line number if it appears at the
c    17         beginning of a line or is immediately preceded by a colon.  Thus,
c    18         for example, the input,
c    19                 Euclid:483: Symbol _627Qui undefined
c    20         might be transcribed as
c    21                 Euclid:483 [line 481 of INCLFILE]: Symbol _627Qui undefined  .)
     22         { Largest SignedInt value (i.e. SignedInt.last). }
     23           ; pervasive const lastSignedInt := 32767                          *     *
     24         { Source line counter modulus. }
     25           ; pervasive const lineNoModulus := 30000                        23    *
     26         { Maximum number of line no. directory entries that will be processed.
c    27         This number does not include the "0th" entry (0,0,''), which is not      *
c    28         present in the line no. directory file.  This limit also in errlist.e }
     29           ; pervasive const maxLineNoDirectoryEntries := 127              25    *
     30                                              { Allows 63 includes }
     31         { Input/output channel number for line no. directory file. }
     32           ; pervasive const lineNoDirectoryFile := 1                      29    *
     33         { Predefined character constants. }
     34           ; pervasive const endOfFile := $$E                              32    *     *
     35           ; pervasive const newLine := $$N                                34    *     *
     36           ; var LineNoResolver                                             *
     37               : module                                                     2
     38                   imports                                                  5
     39                     ( lineNoModulus                                       25
     40                     , IOMode                                               *
     41                     , maxFileNameLength                                    *
     42                     , var IO                                               5
     43                     )
     44                 ; exports                                                  *
     45                     ( ResolveLineNo                                        *
     46                     ; lineNoDirectory                                      *
     47                     , lineNoDirectoryIndex                                 *
     48                     , LineNoDirectoryEntry                                 *
     49                                 { with (fileName, fileNameLength) }
     50                     , LineNoDirectoryTable  ( with (IndexType) )           *
     51                     , inputLineNo                                          *
     52                     )
     53                 { This module handles the resolution of line numbers appearing
c    54                 in the error streams (called "stream line numbers") to their     *     *     *
c    55                 corresponding source input line numbers & file names.  Line
```

```
c   56              numbers are resolved using a table called the line number
c   57              directory.  This table is read from a file created by the
c   58              Screener.  Its values are based upon the placement of include
c   59              clauses in the source program.  The algorithm used to resolve
c   60              line numbers, using the directory, is described in
c   61              Project Euclid Working Paper 42. }
    62              { Type declarations for a line no. directory entry & for the
c   63              line no. directory table. }
    64              ; type LineNoDirectoryEntry                                  48
    65                  = record
    66                      var streamLine : 0 .. lineNoModulus - 1               *     39
    67                    ; var inputLine : 0 .. lineNoModulus - 1                *     39
    68                    ; var fileName                                          *
    69                        : array 1 .. maxFileNameLength of Char              41     5
    70                    ; var fileNameLength : 0 .. maxFileNameLength           *     41
    71                    ;
    72                    end LineNoDirectoryEntry                              64
    73              ; type LineNoDirectoryTable                                   50
    74                  = array 0 .. maxLineNoDirectoryEntries                    29
    75                    of LineNoDirectoryEntry                               72
    76              { Variable declaration for line no. directory. }
    77              ; var lineNoDirectory : LineNoDirectoryTable                  46     73
    78              { Number of entries read into line no. directory. }
    79              ; var noLineNoDirectoryEntries                                *
    80                : 0 .. maxLineNoDirectoryEntries                            29
    81              { Values returned by ResolveLineNo procedure. }
    82              ; var inputLineNo : 0 .. lineNoModulus - 1                    51     39
    83              ; var lineNoDirectoryIndex                                    47
    84                : 0 .. maxLineNoDirectoryEntries                            29
    85              ; procedure ResolveLineNo                                     45
    86                  ( streamLineNo : 0 .. lineNoModulus - 1 )                 *     39
    87                - imports                                                   5
    88                    ( noLineNoDirectoryEntries                            79
    89                    , lineNoDirectory                                     77
    90                    , var inputLineNo                                     82
    91                    , var lineNoDirectoryIndex                            83
    92                    )
    93                ; pre {lineNoDirectory has been initialized}              *
    94                ; post                                                    *
    95                    ( lineNoDirectory                                     89
    96                        ( lineNoDirectoryIndex )                          91
    97                        . streamLine                                      66
    98                      <= streamLineNo                                     86
    99                      and ( lineNoDirectoryIndex                          96
   100                              < noLineNoDirectoryEntries                 88
   101                              -> lineNoDirectory                         95
   102                                   ( lineNoDirectoryIndex + 1 )          99
   103                                   . streamLine                          97
   104                                 > streamLineNo                          86
   105                          )
   106                      and inputLineNo                                     90
   107                          = streamLineNo                                 104
   108                            - lineNoDirectory                            101
   109                                ( lineNoDirectoryIndex )                 102
   110                                . streamLine                             103
   111                            + lineNoDirectory                            101
```

```
   112                                ( lineNoDirectoryIndex )                109
   113                                . inputLine                              67
   114                              )
   115                          { This routine resolves a stream line no. to its
c  116                          corresponding input line no. & file name.  (The file name
c  117                          will be "" for the outer source file.)  The input line
c  118                          no. will be returned in lineNoDirectory(lineNoDirectoryIndex).fileName;
c  119                          its length will be returned in
c  120                          lineNoDirectory(lineNoDirectoryIndex).fileNameLength.
c  121
c  122                          This routine will return possible incorrect results if
c  123                          either of the following conditions holds:
c  124                           1. stream line nos. for the program overflow lineNoModulus,
c  125                              & consequently wrap around;
c  126                           2. the no. of entries in the line no. directory file
c  127                              exceeds maxLineNoDirectoryEntries.  }
   128                        ; begin
   129                          lineNoDirectoryIndex := noLineNoDirectoryEntries  112   100
   130                          ; loop                                             *
   131                            exit when ( lineNoDirectory                      *     *    111
   132                                        ( lineNoDirectoryIndex )             112
   133                                        . streamLine                         110
   134                                      <= streamLineNo                        104
   135                                    )
   136                            ; lineNoDirectoryIndex                           112
   137                              := lineNoDirectoryIndex - 1                    112
   138                            ;
   139                            end loop                                         130
   140                          ; inputLineNo                                      106
   141                            := streamLineNo                                  104
   142                               - lineNoDirectory                             111
   143                                   ( lineNoDirectoryIndex )                  112
   144                                   . streamLine                              110
   145                               + lineNoDirectory                             111
   146                                   ( lineNoDirectoryIndex )                  112
   147                                   . inputLine                               113
   148                          ;
   149                          end ResolveLineNo                                   85
   150                        ; initially {for LineNoResolver}                      *
   151                          imports                                           5
   152                            ( IOMode                                        40
   153                            , var IO                                        5
   154                            , var lineNoDirectory                           111
   155                            , var noLineNoDirectoryEntries                  100
   156                            )
   157                        ; pre ( maxLineNoDirectoryEntries > 0 )             93     29
   158                        ; post {lineNoDirectory(0..noLineNoDirectoryEntries) 94
c  159                            is initialized}
   160                        ; begin
   161                          var k : 1 .. maxLineNoDirectoryEntries + 1        *     157
   162                          ; var inputInt : SignedInt                        *     *
   163                          ; var i : 1 .. maxFileNameLength                  *     41
   164                            {Set 0th directory entry to (0,0,"").}
   165                          ; lineNoDirectory ( 0 ) . streamLine := 0         154   110
   166                          ; lineNoDirectory ( 0 ) . inputLine := 0          165   113
   167                          ; lineNoDirectory ( 0 ) . fileNameLength := 0     166    70
```

```
168                     ; IO . Open ( lineNoDirectoryFile , inFile )                S     *     32     *
169                     ; k := 1                                                   161
170                     ; loop                                                     130
171                         IO . ReadInt                                            S     *
172                           ( lineNoDirectoryFile , inputInt )                   168   162
173                       ; exit when ( IO . EndFile                               131   131     S     *
174                                       ( lineNoDirectoryFile )                  172
175                                   )
176                       ; lineNoDirectory ( k ) . streamLine := inputInt         167   169   165   172
177                       ; IO . ReadInt                                            S     171
178                           ( lineNoDirectoryFile , inputInt )                   174   176
179                       ; lineNoDirectory ( k ) . inputLine := inputInt          176   176   166   178
180                       ; IO . ReadInt                                            S     177
181                           ( lineNoDirectoryFile , inputInt )                   178   179
182                       ; lineNoDirectory ( k ) . fileNameLength                 179   179   167
183                         := inputInt                                            181
184                       ; if inputInt > 0                                        183
185                         then i := 1                                            163
186                         ; loop                                                 170
187                             IO . ReadChar                                       S     *
188                               ( lineNoDirectoryFile , lineNoDirectory          181   182
189                                   ( k )                                        182
190                                   . fileName                                    68
191                                   ( i )                                        185
192                               )
193                           ; exit when i = inputInt                             173   173   191   184
194                           ; i := i + 1                                         193    *
195                           ;
196                           end loop                                             186
197                         ;
198                         end if
199                       ; assert                                                  *
200                           ( not IO . EndFile ( lineNoDirectoryFile ) )          S     173   188
201                       ; k := k + 1                                             189    *
202                       ; exit when ( k > maxLineNoDirectoryEntries )            193   193   201   157
203                       ;
204                       end loop                                                 196
205                     ; noLineNoDirectoryEntries := k - 1                        155   202
206                     ; IO . Close ( lineNoDirectoryFile )                        S     *     200
207                     ;
208                     end (initially)
209                 ;
210                 end module (LineNoResolver)                                     2
211             ; initially                                                       150
212                 imports ( var IO , var LineNoResolver )                         5     S     36
213               ; pre ( lineNoModulus <= ( lastSignedInt ) div 10 * 10 )        157    39     23
214               ; begin
215                   var inputChar : Char     { Current input char. }              *     S
216                 ; var accumulatedNo : 0 .. lastSignedInt { Value of digits read                *   213
c   217                     thus far in stream line no. }
218                 ; var inStreamLineNoContext : Boolean { Indicates whether next      *     S
c   219                     char might be part of stream line no. }
220                 ; var k : 1 .. maxFileNameLength                               205    41
221                 ; accumulatedNo := 0                                           216
222                 ; inStreamLineNoContext := true                                218     S
223                 ; loop IO . GetChar ( inputChar )                              204     S     *   215
```

```
224                 ; exit when inputChar = endOfFile                          202   202   223   34
225                 ; if inStreamLineNoContext                                 222
226                   then
227                     if inputChar >= $0 and inputChar <= $9                   224    *
228                     then
229                       if accumulatedNo <= lastSignedInt div 10 - 1           221   213
230                       then
231                         accumulatedNo                                        221
232                         := accumulatedNo                                     221
233                            * 10
234                            + ( Char . Ord ( inputChar )                        S     S    227
235                                - Char . Ord ( $0 )                             S     S
236                              )
237                       ;
238                       else
239                               ( Number is too big.  Forget about resolving it.)
240                         accumulatedNo := 0                                   221
241                       ; inStreamLineNoContext := false                       225     S
242                       ;
243                       end if
244                     ;
245                     else
246                       if accumulatedNo not = 0                               240
247                       then
248                         LineNoResolver . ResolveLineNo                       212    85
249                           ( accumulatedNo )                                  246
250                       ; IO . PutString ( ' (' )                               S     *
251                                 ( Output resolved line no. & file name. )
252                       ; IO . PutString ( 'line ' )                            S   250
253                       ; IO . PutInt ( LineNoResolver . inputLineNo )          S     *   212   106
254                       ; if LineNoResolver . lineNoDirectory                  253   188
255                              ( LineNoResolver . lineNoDirectoryIndex )       254   112
256                              . fileNameLength                                182
257                            > 0
258                         then k := 1                                          220
259                         ; IO . PutString ( ' of ' )                           S   252
260                         ; loop                                               223
261                             IO . PutChar                                      S     *
262                               ( LineNoResolver . lineNoDirectory             255   254
263                                   ( LineNoResolver                           262
264                                       . lineNoDirectoryIndex                 255
265                                   )
266                                   . fileName                                 190
267                                   ( k )                                      258
268                               )
269                           ; exit when ( k                                    224   224   267
270                                         >= LineNoResolver                    263
271                                              . lineNoDirectory               262
272                                              ( LineNoResolver                263
273                                                  .
274                                                    lineNoDirectoryIndex      264
275                                              )
276                                              . fileNameLength                256
277                                       )
278                           ; k := k + 1                                       269     *
279                           ;
```

```
280                             end loop                                    260
281                           ;
282                           end if
283                         ; IO . PutChar ( $) )                           5    261
284                         ; accumulatedNo := 0                            249
285                         ;
286                         end if
287                       ; inStreamLineNoContext := false                  241     S
288                       ;
289                       end if
290                   ;
291                   end if
292                 ; if inputChar = $: or inputChar = newLine              234     *     35
293                   then inStreamLineNoContext := true                    287     S
294                   ;
295                   end if
296                 ; IO . PutChar ( inputChar )                            S    283    292
297                 ;
298                 end loop                                                280
299               ;
300               end (initially)
301           ;
302           end module (InteractiveLineNoResolver}                        210
303     ; initially begin end                                               211
304     ;
305     end module (WORLD}                                                  302
306 ;
307
```

cross reference * is def = is asg

```
-A-
 accumulatedNo          216*  221=  229   231=  232   240=  246*  249*  284=
 assert                 199*

-B-
 Boolean                218

-C-
 Char                    69   215   234*  235*
 Close                  206*

-E-
 E                       34*
 EndFile                173*  200*
 endOfFile               34=  224
 exit                   131   173*  193*  202*  224*  269*
 exports                 44*

-F-
 false                  241*  287*
 fileName                68*  190*  266*
 fileNameLength          70*  167=  182=  256*  276*

-G-
 GetChar                223*

-I-
 I                      163*  185=  191*  193*  194=  194*
 Imports                  5    38    87   151   212*
 inFile                 168*
 initially              150   211*  303*
 InputChar              215*  223*  224*  227*  227   234*  292*  292   296*
 InputInt               162*  172*  176*  178*  179*  181*  183*  184*  193
 InputLine               67*  113*  147   166=  179=
 InputLineNo             51*   82*   90*  106*  140=  253*
 InStreamLineNoContext  218*  222=  225*  241=  287=  293=
 InteractiveLineNoResol-   3
   ver
 IO                       5*   42*  153*  168*  171*  173*  177*  180*  187*  200*  206*  212*  223*  250*  252*  253*
                        259*  261*  283*  296*
 IOMode                  40*  152

-K-
 k                      161*  169=  176*  179*  182*  189*  201=  201*  202*  205*  220*  258=  267*  269*  278=  278*

-L-
 lastSignedInt           23=  213*  216   229
 line                    54*
 lineNoDirectory         46*   77*   89*   95*  101*  108   111*  131   142   145   154*  165*  166*  167*  176*  179*
                        182*  188*  254*  262*  271
 LineNoDirectoryEntry    48*   64*   72*   75
 lineNoDirectoryFile     32=  168*  172*  174*  178*  181*  188*  200*  206*
 lineNoDirectoryIndex    47*   83*   91*   96*   99*  102*  109*  112*  129=  132   136=  137   143   146   255*  264*
```

```
                          274*
 LineNoDirectoryTable      50*   73*   77
 lineNoModulus             25=   39*   66    67    82    86   213*
 LineNoResolver            36*  212*  248   253*  254*  255*  262*  263*  270   272*
 loop                     130   139   170*  186*  196*  204*  223*  260*  280*  298*

-M-
 maxFileNameLength         41*   69    70   163   220
 maxLineNoDirectoryEntr-   29=   74    80    84   157*  161   202*
   ies
 module                     2     4    37   210*  302*  305

-N-
 N                         35*
 newLine                   35=  292
 noLineNoDirectoryEntri-   79*   88*  100*  129   155*  205=
   es
 numbers                   54*

-O-
 Open                     168*
 Ord                      234*  235*

-P-
 pervasive                 23*   25*   29*   32*   34*   35*
 post                      94*  158*
 pre                       93*  157*  213*
 PutChar                  261*  283*  296*
 PutInt                   253*
 PutString                250*  252*  259*

-R-
 ReadChar                 187*
 ReadInt                  171*  177*  180*
 ResolveLineNo             45*   85*  149   248

-S-
 SignedInt                162
 stream                    54*
 streamLine                66*   97*  103*  110*  133   144   165=  176=
 streamLineNo              86*   98   104*  107   134   141

-T-
 th                        27*
 true                     222*  293

-W-
 when                     131   173*  193*  202*  224*  269*
 WORLD                      1*

end xref   63 identifiers   297 total references
  99 collisions.
```

Appendix 3

Euclid Pretty-Printer code

```
     1
     2 "*************************************************
c    3  *                                                *
c    4  *     EUCLID EPP              VERSION 1.0        *
c    5  *     Wun Jen Lin             02/28/83           *
c    6  *                                                *
c    7  *************************************************"
     8 "% PASCAL32 := TRUE "
     9 "% PASCAL := NOT PASCAL32"
    10 "% ASCII := TRUE"
    11 "% EBCDIC := NOT  ASCII"
    12 "% VAX := FALSE"
    13 "% PREP := TRUE"
    14 "% SINGLEPRECISION := TRUE"
    15 "% MULTPRECISION := NOT SINGLEPRECISION"
    16 "%IF PASCAL32"
    17 "###########
c   18 #  PREFIX  #
c   19 ###########"
    20   type integer32 = integer                                          *     S
    21
    22 ; const maxint32 = maxint                                          *     S
    23
    24 "% IF MULTPRECISION"
    25
    26 ; type integer = shortinteger                                      S     S
    27
    28 ; const maxint = maxshortint                                       S     S
    29
    30 "% END MULTPRECISION"
    31
    32 ; const nl = '(:10:)'                                             *
    33   ; ff = '(:12:)'                                                 *
    34   ; cr = '(:13:)'                                                 *
    35   ; em = '(:25:)'                                                 *
    36
    37 ; const pagelength = 512                                          *
    38
    39 ; type page = array [ 1 .. pagelength ] of char                   S    37     S
    40
    41 ; const linelength = 132                                          *
    42
    43 ; type line = array [ 1 .. linelength ] of char                   *    41     S
    44
    45 ; const idlength = 12                                             *
    46
    47 ; type identifier = array [ 1 .. idlength ] of char              *    45     S
    48
    49 ; type pfxfile = 1 .. 5                                          *
    50
    51 ; type filekind = ( empty , scratch , ascii , seqcode , concode ) *     S    *    *    *    *
    52
    53 ; type fileattr                                                   *
    54     = record
    55         kind : filekind                                          *    51
```

```
    ; addr : integer                                                    *     S
    ; protected : boolean                                              *     S
    ; notused : array [ 1 .. 5 ] of integer                            *     S
    end

; type iodevice                                                        *
  = ( typedevice , diskdevice , tapedevice , printdevice              *     *     *     *
    , carddevice )                                                     *

; type iooperation = ( input , output , move , control )              *     S     S     *     *

; type ioarg = ( writeeof , rewind , upspace , backspace )            *     *     *     *     *

; type ioresult                                                       *
  = ( complete , intervention , transmission , failure , endfile      *     *     *     *     *
    , endmedium , startmedium )                                        *     *

; type ioparam                                                        *
  = record
      operation : iooperation                                        *    65
    ; status : ioresult                                              S    69
    ; arg : ioarg                                                    *    67
    end

; type taskkind = ( inputtask , jobtask , outputtask )               *     *     *     *

; type argtag = ( niltype , booltype , inttype , idtype , ptrtype )  *     *     *     *     *     *

; type pointer = ^ boolean                                           *     S

; type passptr = ^ passlink                                          *     *

; type passlink                                                      86
  = record
      options : set of char                                          *     S
    ; filler1 : array [ 1 .. 7 ] of integer                          *     S
    ; filler2 : boolean                                              *     S
    ; resetPoint : integer                                           *     S
    ; filler3 : array [ 1 .. 11 ] of pointer                         *    84
    end

; type argtype                                                       *
  = record
      case tag : argtag                                              *    82
      of niltype , booltype : ( bool : boolean )                     82    82     *     S
      ; inttype : ( int : integer )                                  82     *     S
      ; idtype : ( id : identifier )                                 82     *    47
      ; ptrtype : ( ptr : passptr )                                  82     *    86
    end

; const maxarg = 10                                                  *

; type arglist = array [ 1 .. maxarg ] of argtype                    *   106    97

; type argseq = ( inp , out )                                        *     *     *
```

```
112 ; type progresult                                                          *
113     = ( terminated , overflow , pointererror , rangeerror                  *     *     *     *
114       , varianterror , heaplimit , stacklimit , codelimit , timelimit      *     *     *     *     *
115       , callerror )                                                        *
116
117 ; procedure pfxread ( var c : char )                                        *     *     S
118
119 ; procedure pfxwrite ( c : char )                                          *   117     S
120
121 ; procedure open                                                           *
122     ( f : pfxfile ; id : identifier ; var found : boolean )                *    49   102    47     *     S
123
124 ; procedure close ( f : pfxfile )                                          *   122    49
125
126 ; procedure pfxget                                                         *
127     ( f : pfxfile ; p : integer ; var block : univ page )                 124    49     *     S     *     S
128
129 ; procedure put ( f : pfxfile ; p : integer ; var block : univ page )       S   127    49   127     S   127     S
130
131 ; function length ( f : pfxfile ) : integer                                *   129    49     S
132
133 ; procedure mark ( var top : integer )                                     S     *     S
134
135 ; procedure release ( top : integer )                                      S   133     S
136
137 ; procedure identify ( header : line )                                     *     *    43
138
139 ; procedure accept ( var c : char )                                        *   119     S
140
141 ; procedure display ( c : char )                                           *   139     S
142
143 ; procedure notused                                                       58
144
145 ; procedure notused2                                                      *
146
147 ; procedure notused3                                                      *
148
149 ; procedure notused4                                                      *
150
151 ; procedure notused5                                                      *
152
153 ; procedure notused6                                                      *
154
155 ; procedure notused7                                                      *
156
157 ; procedure notused8                                                      *
158
159 ; procedure notused9                                                      *
160
161 ; procedure notused10                                                     *
162
163 ; procedure run                                                           S
164     ( id : identifier                                                   122    47
165     ; var param : arglist                                                *   108
166     ; var line : integer                                                43     S
167     ; var result : progresult                                            *   112
```

```
   168      )
   169
   170 ; program p ( param : line )                                    129   165   166
   171 "% END PASCAL32"
   172 "% IF PASCAL"
   173      program ppp ( input , output , initfile , summary )          *     S     S     *     *
   174
   175 "% END"
   176
   177      ;
   178 include 'epp2.scd'                                              *
   179 " Declarations used by semantic routines and interpreter "
   180        { Primitive table Operations:
c  181
c  182
c  183          These will remain the same independent of the
c  184          S/SL program and its semantic actions. }
   185       const firsttableOperation = 0                             *
   186
   187     ; const firstPrimitiveOperation = 0                         *
   188
   189     ; const oCall = 0                                           *
   190
   191     ; const oReturn = 1                                         *
   192
   193     ; const oRepeat = 2                                         *
   194
   195     ; const oMerge = 3                                          *
   196
   197     ; const oInput = 4                                          *
   198
   199     ; const oInputChoice = 5                                    *
   200
   201     ; const oEmit = 6                                           *
   202
   203     ; const oError = 7                                          *
   204
   205     ; const oChoiceEnd = 8                                      *
   206
   207     ; const lastPrimitiveOperation = 8                          *
   208
   209         { Semantic Operations:
c  210
c  211          These will be different for each pass.  The
c  212          semantic operations are implemented in the
c  213          semantic module. }
   214
   215     ; const firstSemanticOperation = 14                         *
   216
   217     ; const firstSimpleOperation = 14                           *
   218
   219     ; const lastSimpleOperation = 20                            *
   220
   221     ; const firstParameterizedOperation = 21                    *
   222
   223     ; const lastParameterizedOperation = 26                     *
```

```
224
225     ; const firstChoiceOperation = 26                                   *
226
227     ; const lastChoiceOperation = 26                                    *
228
229     ; const firstEmittingOperation = 26                                 *
230
231     ; const lastEmittingOperation = 26                                  *
232
233     ; const lastSemanticOperation = lastEmittingOperation               *   231
234
235     ; const lasttableOperation = lastSemanticOperation                  *   233
236
237     ; const dummyLine = 0                                               *
238
239     ; const setSize = 7                                                 *
240
241     ; type SetArray = array [ 0 .. setSize ] of shortinteger            *   239     S
242       ; symboltyp = 0 .. 127                                            *
243       ; setofsymbol = set of symboltyp                                  *   242
244
245     ; type CallStackEntry                                               *
246         = record
247             sett : setofsymbol                                          *   243
248                                 { Dynamic set of syntax recovery tokens}
249           ; returnAddress : 0 .. tableSize                              *     *
250           ;
251           end {CallStackEntry}
252
253     ; const callStackSize = 127                                         *
254
255     ; const maxLineNumber = 29999                                       *
256
257         { Error Output Interface }
258
259     ; const firstErrorCode = 0                                          *
260
261     ; const eNoError = 0                                                *
262
263     ; const eEndOfErrors = 1                                            *
264
265     ; const firstTrueErrorCode = 2                                      *
266
267     ; const firstParserErrorCode = 21                                   *
268
269     ; const eSyntaxError = 21                                           *
270
271     ; const firstFatalErrorCode = 22                                    *
272
273     ; const ePrematureEndOfFile = 22                                    *
274
275     ; const eCallStackOverflow = 23                                     *
276
277     ; const eExtraneousProgramText = 24                                 *
278
279     ; const ePosStackOverflow = 25                                      *
```

```
280
281     ; const ePosStackUnderflow = 26                                    *
282
283     ; const eMarkStackOverflow = 27                                    *
284
285     ; const eMarkStackUnderflow = 28                                   *
286
287     ; const lastErrorCode = 28                                         *
288
289     ; type ErrorCodes = firstErrorCode .. lastErrorCode                *   259   287
290
291     ; const maxErrors = 20                                             *
292
293         { Input Interface }
294
295     ; const firstInputToken = - 1                                      *
296
297     ; const tSyntaxError = - 1                                         *
298
299     ; const lastInputToken = tXor                                      *     *
300
301     ; const leftmargin = 1                                             *
302       ; rightmargin = 72                                               *
303       ; MaxPosStack = 127                                              *
304       ; MaxMarkStack = 127                                             *
305       ; shiftpos = 36                                                  *
306       ; indentamt = 2                                                  *
307       ; fraglength = 12                                                *
308       ; rwmax = 80                                                     *
309       "% IF ASCII"
310       ; ht = 9 "HORIZONTAL TAB"                                        *
311       ; ff = 12 "FORM FEED"                                           33
312       "% END ASCII"
313       "% IF EBCDIC"
314       ; ht = 5 "HORIZONTAL TAB"                                      310
315       ; ff = 12 "FORM FEED"                                          311
316       "% END EBCDIC"
317       ; multmct                                                        *
318         = 2 "NUMBER OF INTEGERS IN A MULTIPLE PRECISION INTEGER"
319
320     ; type positiontyp = integer                                       *     S
321       "% IF SINGLEPRECISION"
322       ; multint = integer                                              *     S
323       "% END SINGLEPRECISION"
324       "% IF MULTPRECISION"
325       ; multint = array [ 1 .. multmct ] of integer                  322   317     S
326         "MULTMCT IS MOST SIGNIFICANT"
327       "% END MULTPRECISION"
328       ; setofchar = set of char                                        *     S
329       ; linerecptr = ^ linerec                                         *     *
330       ; linerec                                                      329
331         = record
332             dlink , rlink : linerecptr                                 *     *   329
333           ; tokno , aggsize : multint                                  *     *   325
334           ; symbol : symboltyp                                         *   242
335           ; startpos : positiontyp                                     *   320
```

```
    336           ; length : positiontyp                                          131    320
    337           ; newline : boolean                                               *      S
    338           ; skip : boolean                                                  *      S
    339           ; sepafter : boolean                                              *      S
    340           ; moveable : boolean                                              *      S
    341           ; text                                                            S
    342             : "% IF PASCAL"
    343               packed "% END " array [ 1 .. fraglength ] of char             *    307      S
    344             end
    345         ; markrec = record linep : linerecptr ; tokno : multint end         *      *    329    333    325
    346         ; parproc                                                           *
    347           = ( parconstant , parstc , parstcval , parstcelem                 *      *      *      *
    348             , parvariable , parsetelem , parfactelem , parfactor            *      *      *      *
    349             , parterm , parsmpexpr , parexpr , paractparam                  *      *      *      *
    350             , parcompoundstmt , parxstmt , parlcase , parstmt , parfparm    *      *      *      *      *
    351             , parsmptype , parproghead , parpfhead , parfidlist             *      *      *      *
    352             , partype , parsyscomp , partypenamer , parblock                *      *      *      *
    353             , parprogram )                                                  *
    354         ; string12                                                          *
    355           =  "% IF PASCAL "
    356             packed "% END " array [ 1 .. 12 ] of char                      343      S
    357         ; stringType = array [ 1 .. 100 ] of char                           *      S
    358         "% IF PASCAL32"
    359         ; logicalunitno = 0 .. 7                                            *
    360         ; filestatus = ( closed , opforin , opforout )                      *      *      *      *
    361         ; pageptr = ^ page                                                  *      S
    362         ; text                                                              S
    363           = record
    364               fno : logicalunitno                                           *    359
    365             ; pageno , bufp : integer                                       *      *      S
    366             ; status : filestatus                                           S    360
    367             ; eoinf , eoff : boolean                                        *      *      S
    368             ; bufferp : pageptr                                             *    361
    369             ; currentchar : char                                            *      S
    370             end
    371
    372       ;
    373         "% END PASCAL32"
    374    { The Syntax/Semantic table }
    375           { S/SL Interpreter State }
    376         var processing : Boolean                                            *      S
    377         ; tracing : Boolean                                                 *      S
    378         ; semanticizing : Boolean                                           *      S
    379         ; displaySyntaxErrors : Boolean                                     *      S
    380         ; tablePointer : 0 .. tableSize                                     *    249
    381         ; operation : firsttableOperation .. lasttableOperation            75    185    235
    382           { Line Counter }
    383         ; sourceLineNo : 0 .. maxLineNumber                                 *    255
    384         ; keys : setofsymbol                                                *    243
    385         ; Lmark : markrec                                                   *    345
    386         ; choicetagMatched : boolean                                        *      S
    387           { The Call Stack:
c   388
c   389           The Call Stack implements Syntax/Semantic
c   390           Language procedure call and return.  In addition,
c   391           the Call Stack contains dynamic error recovery sets.
```

```
c  392          Each time an oCall operation is executed,
c  393          the table return address and the reachable set
c  394          for the oCall operator are pushed onto the
c  395          Call Stack.  When an oReturn is executed, the
c  396          return address and the reachable set are popped from the
c  397          Stack. An oReturn executed when the Call stack is
c  398          empty terminates interpretation.)
   399        ; callStackTop : 0 .. callStackSize                          *   253
   400        ; callStack : array [ 0 .. callStackSize ] of CallStackEntry *   253   245
   401          { Parameterized And Choice Semantic Operation Values:
c  402
c  403          These are used to hold the decoded parameter value to
c  404          a parameterized semantic operation and the result
c  405          value returned by a choice semantic operation
c  406          respectively.                                    }
   407        ; parameterValue : integer                                   *     S
   408        ; resultValue : integer                                      *     S
   409        ; gError : 0 .. maxErrors                                    *   291
   410        " NEWLENGTH THRU NEWP USED BY INSYMBOL AND SUBROUTINES "
   411        ; newlength : integer                                        *     S
   412        ; pos , newstartpos , firstdotpos : positiontyp              *     *     *   320
   413        ; innewline : boolean                                        *     S
   414        ; rbracksw , ellpsissw : boolean                             *     *     S
   415        ; cmntstate : ( nocomment , bracecomment )                   *     *     *
   416        ; newsymbol : symboltyp                                      *   242
   417        ; newp : linerecptr                                          *   329
   418        ; i : integer                                                *     S
   419        ; gfirstp , gtrailp , gcompp , gcomptrailp , gcurp , gnextp  *,    *     *     *     *     *
   420          , gfreep                                                   *
   421          : linerecptr                                             329
   422        ; gcursy : symboltyp                                         *   242
   423        ; gcurpos , gcurposnc , gpos , outpos : positiontyp          *     *     *     *   320
   424        ; gmark : markrec                                            *   345
   425        ; gtokno , outtokno , nltokno : multint                      *     *     *   325
   426        ; gaggsize : multint                                         *   325
   427        ; PosStack : array [ 1 .. MaxPosStack ] of positiontyp       *   303   320
   428        ; MarkStack : array [ 1 .. MaxMarkStack ] of markrec         *   304   345
   429        ; PosStackTop : 0 .. MaxPosStack                             *   303
   430        ; MarkStackTop : 0 .. MaxMarkStack                           *   304
   431        "% IF MULTPRECISION"
   432        ; linespace , multrightmargin : multint                      *     *   325
   433        "% END MULTPRECISION"
   434        ; rwptr : array [ 1 .. 13 ] of 1 .. rwmax                    *   308
   435        ; rw                                                         *
   436          : array [ 1 .. rwmax ]                                   308
   437            of record
   438                 string : string12                                 *   354
   439               ; symbol : symboltyp                               334   242
   440               end
   441        ; rwct : integer                                             *     S
   442        ; letters , octaldigits , digits , hexdigits , lexchars      *     *     *     *     *
   443          , throwaways                                               *
   444          : setofchar                                              328
   445        ; lineoutct , linerecct , tokenct , skipct : integer         *     *     *     *     S
   446        "% IF PASCAL"
   447        ; initfile , summary : text                                173   173     S
```

```
448       "% END PASCAL"
449       "% IF PASCAL32"
450       ; input , output , initfile , summary : text                    S     S   447   447     S
451
452     ; procedure breakpnt                                              *
453       ; extern
454
455     ; procedure get ( var f : text )                                  *   131     S
456
457       ; begin
458           with f                                                    455
459           do if not eoff                                            367
460             then if fno in [ 0 , 1 , 3 , 4 , 5 , 6 , 7 ]            364
461               then begin
462                 case fno                                            364
463                 of 0 : accept ( currentchar )                       139   369
464                 ; 1 : pfxread ( currentchar )                       117   369
465                 ; 3 , 4 , 5 , 6
466                   : if status = opforin                               S   360
467                     then begin
468                       if bufp > pagelength                          365    37
469                       then begin
470                         pfxget ( fno - 2 , pageno , bufferp ^ )     126   364   365   368
471                       ; pageno := succ ( pageno )                   365     S   365
472                       ; bufp := 1                                   365
473                       end
474                     ; currentchar := bufferp ^ [ bufp ]             369   368   365
475                     ; bufp := succ ( bufp )                         365     S   365
476                     end
477                     else currentchar := ' '                         369
478               end " CASE "
479             ; if currentchar = em                                   369    35
480               then begin eoff := true ; currentchar := ' ' end      367     S   369
481               else if currentchar = nl                              369    32
482                 then begin eoinf := true ; currentchar := ' ' end   367     S   369
483                 else eoinf := false                                 367     S
484             end
485             else
486           else currentchar := ' '                                   369
487       end " READ "
488
489     ; procedure open                                                121
490         ( var f : text ; lu : logicalunitno ; mode : filestatus )   455     S     *   359     *   360
491
492       ; begin
493           with f                                                    490
494           do begin
495             if lu in [ 3 , 4 , 5 , 6 , 7 ]                          490
496             then begin
497               pageno := 1                                           365
498             ; if mode = opforin                                     490   360
499               then bufp := pagelength + 1                           365    37
500               else if mode = opforout then bufp := 1                490   360   365
501             ; status := mode                                        S     490
502             ; new ( bufferp )                                       S     368
503             end
```

```
504           ; eoff := false                                          367     5
505           ; eoinf := false                                         367     5
506           ; fno := lu                                              364   490
507           end
508         ; get ( f )                                                455   490
509         end " OPEN "
510
511       ; function inputwindow ( var f : text ) : char                 *   490     5     5
512
513         ; begin
514             if f . currentchar in throwaways                       511   369   443
515             then inputwindow := ' '                                511
516             else inputwindow := f . currentchar                    511   511   369
517           end " INPUTWINDOW "
518
519       ; function eof ( var f : text ) : boolean                      5   511     5     5
520
521         ; begin eof := f . eoff end                                  5   519   367
522
523       ; function eoln ( var f : text ) : boolean                     5   519     5     5
524
525         ; begin eoln := f . eoinf end                                5   523   367
526
527       ; procedure write ( var f : text ; c : char )                  5   523     5   141     5
528
529         ; begin
530             with f                                                 527
531             do case fno                                            364
532               of 0 : display ( c )                                 141   527
533               ; 2 : pfxwrite ( c )                                 119   527
534               ; 3 , 4 , 5 , 6 , 7
535                 : if status = opforout                             5   360
536                   then begin
537                     if bufp > pagelength                           365    37
538                     then begin
539                       put ( fno - 2 , pageno , bufferp ^ )          5   364   365   368
540                     ; pageno := succ ( pageno )                    365     5   365
541                     ; bufp := 1                                    365
542                     end
543                   ; bufferp ^ [ bufp ] := c                        368   365   527
544                   ; bufp := succ ( bufp )                          365     5   365
545                   end
546             end
547         end " WRITE "
548
549       ; procedure writeln ( var f : text )                           5   527     5
550
551         ; begin write ( f , nl ) end                                 5   549    32
552
553       ; procedure close ( var f : text )                           124   549     5
554
555         ; begin
556             with f                                                 553
557             do if fno = 2                                          364
558               then pfxwrite ( em )                                 119    35
559               else if ( fno in [ 3 , 4 , 5 , 6 ] )                 364
```

```
560                     and ( status = opforout )                                S   360
561                 then begin
562                   write ( f , em )                                           S   553    35
563                 ; put ( fno - 2 , pageno , bufferp ^ )                       S   364   365   368
564                 ; status := closed                                           S   360
565                 end
566           end " CLOSE "
567
568         "% END PASCAL32"
569       "% IF PASCAL "
570
571       ; function inputwindow ( var f : text ) : char                        511   553     S     S
572
573         ; begin
574           ; if f ^ in throwaways                                             571   443
575             then inputwindow := ' '                                          571
576             else inputwindow := f ^                                          571   571
577           end
578
579 "% END PASCAL "
580 "% IF MULTPRECISION"
581 " PROCEDURES FOR MULTIPLE PRECISION ARITHMETIC"
582
583       ; procedure multzero ( var result : multint )                          *   167   325
584
585         ; var i : 1 .. multmct                                               418   317
586
587         ; begin for i := 1 to multmct do result [ i ] := 0 end               585   317   583   585
588
589       ; procedure multplint                                                  *
590           ( mop : multint ; iop : integer ; var result : multint )           *   325     *     S   583   325
591
592         ; var i , carryin , work : integer                                   585     *     *     S
593           ; tempresult : multint                                             *   325
594
595         ; begin
596             carryin := iop                                                   592   590
597           ; for i := 1 to multmct                                            592   317
598             do begin
599               if mop [ i ] > 0                                               590   592
600               then if ( mop [ i ] - maxint ) > ( - carryin )                 590   592     S   592
601                 then begin "PARANIOD CODING"
602                   work := mop [ i ] - maxint                                 592   590   592     S
603                 ; work := work - 2                                           592   592
604                 ; work := work + carryin                                     592   592   592
605                 ; tempresult [ i ] := work - maxint                          593   592   592     S
606                 end
607                 else tempresult [ i ] := mop [ i ] + carryin                 593   592   590   592   592
608               else tempresult [ i ] := mop [ i ] + carryin                   593   592   590   592   592
609             ; if mop [ i ] < 0                                             590   592
610               then if mop [ i ] + carryin >= 0                             590   592   592
611                 then carryin := 1                                          592
612                 else carryin := 0                                          592
613               else carryin := 0                                            592
614             end
615           ; result := tempresult                                           590   593
```

```
616         end " MULTPLINT "
617
618       ; procedure multsubtr                                          *
619           ( op1 , op2 : multint ; var result : multint )             *     *   325   590   325
620
621         ; var i , work : integer                                    592   592     S
622           ; borrowin , borrowout : boolean                            *     *     S
623           ; tempresult : multint                                    593   325
624
625         ; begin
626             borrowin := false                                       622     S
627           ; for i := 1 to multmct                                   621   317
628             do begin
629               if ( op1 [ i ] >= 0 ) and ( op2 [ i ] < 0 )           619   621   619   621
630               then begin
631                 borrowout := true                                   622     S
632               ; if op1 [ i ] - maxint > op2 [ i ]                   619   621     S   619   621
633                 then begin "PARANOID CODING"
634                   work := op1 [ i ] - maxint                        621   619   621     S
635                 ; work := work - 1                                  621   621
636                 ; work := work - op2 [ i ]                          621   621   619   621
637                 ; work := work - maxint                             621   621     S
638                 ; tempresult [ i ] := work - 1                      623   621   621
639                 end
640                 else tempresult [ i ] := op1 [ i ] - op2 [ i ]      623   621   619   621   619   621
641               end
642               else if ( op1 [ i ] < 0 ) and ( op2 [ i ] >= 0 )      619   621   619   621
643                 then begin
644                   borrowout := false                                622     S
645                 ; if op1 [ i ] + maxint < op2 [ i ] - 1             619   621     S   619   621
646                   then begin "MORE PARANOID"
647                     work := op1 [ i ] + maxint                      621   619   621     S
648                   ; work := work + 1                                621   621
649                   ; work := work - op2 [ i ]                        621   621   619   621
650                   ; work := work + maxint                           621   621     S
651                   ; tempresult [ i ] := work + 1                    623   621   621
652                   end
653                   else tempresult [ i ] := op1 [ i ] - op2 [ i ]    623   621   619   621   619   621
654                 end
655                 else begin
656                   borrowout := op1 [ i ] < op2 [ i ]                622   619   621   619   621
657                 ; tempresult [ i ] := op1 [ i ] - op2 [ i ]         623   621   619   621   619   621
658                 end
659             ; if borrowin                                           622
660               then if tempresult [ i ] < - maxint                   623   621     S
661                 then tempresult [ i ] := maxint                     623   621     S
662                 else begin
663                   if tempresult [ i ] = 0                           623   621
664                   then borrowout := true                            622     S
665                 ; tempresult [ i ] := pred ( tempresult [ i ] )     623   621     S   623   621
666                 end
667             ; borrowin := borrowout                                 622   622
668             end
669           ; result := tempresult                                    619   623
670           end " MULTSUBTR "
671
```

```
672     ; function multge ( op1 , op2 : multint ) : boolean                    *   619   619   325     S
673
674       ; var work : multint                                                621   325
675
676       ; begin
677           multsubtr ( op1 , op2 , work )                                  618   672   672   674
678         ; multge := work [ multmct ] >= 0                                 672   674   317
679         end " MULTGE "
680
681     ; function multgt ( op1 , op2 : multint ) : boolean                    *   672   672   325     S
682
683       ; var work : multint                                                674   325
684
685       ; begin
686           multplint ( op2 , 1 , work )                                    589   681   683
687         ; multsubtr ( op1 , work , work )                                 618   681   683   683
688         ; multgt := work [ multmct ] >= 0                                 681   683   317
689         end " MULTGT "
690
691 "% END MULTPRECISION"
692
693     ; procedure int2string ( fint : integer ; var fstring : string12 )     *     *     S     *   354
694
695       ; var iint , digit : integer                                        *     *     S
696
697       ; begin
698           iint := fint                                                    695   693
699         ; i := 12                                                         621
700         ; fstring := '            '                                       693
701         ; if iint = 0                                                     695
702           then fstring [ 12 ] := '0'                                      693
703           else while ( iint <> 0 ) and ( i >= 1 )                         695   621
704             do begin
705               digit := iint mod 10                                        695   695
706             ; fstring [ i ] := chr ( ord ( '0' ) + digit )                693   621     S     S   695
707             ; i := pred ( i )                                             621     S   621
708             ; iint := iint div 10                                         695   695
709             end
710         end " INT2STRING "
711
712     ; procedure reportstat ( fnum : integer ; fstr : string12 )            *     *     S     *   354
713
714       ; var inumstr : string12                                            *   354
715         ; i : integer                                                     621     S
716
717       ; begin
718           int2string ( fnum , inumstr )                                   693   712   714
719         ; for i := 1 to 12 do write ( summary , inumstr [ i ] )           715     S   450   714   715
720         ; write ( summary , ' ' )                                           S   450
721         ; for i := 1 to 12 do write ( summary , fstr [ i ] )              715     S   450   712   715
722         ; writeln ( summary )                                               S   450
723         end " REPORTSTAT "
724
725     ; procedure DisplayInteger ( Number : integer )                        *     *     S
726
727       ; var i : integer                                                   715     S
```

```
728         ; inumstr : string12                                          714   354
729
730       ; begin
731           int2string ( Number , inumstr )                             693   725   728
732         ; i := 1                                                      727
733         ; while ( i <= 12 ) and ( inumstr [ i ] = ' ' )               727   728   727
734           do i := succ ( i )                                          727     S   727
735         ; display ( ' ' )                                             141
736         ; while i <= 12                                               727
737           do begin
738             display ( inumstr [ i ] )                                 141   728   727
739           ; i := succ ( i )                                           727     S   727
740           end
741         ; display ( ' ' )                                             141
742         end { DisplayInteger }
743
744     ; procedure DisplayString ( s : StringType )                        *     *     *
745
746       ; var i : integer                                               727     S
747
748       ; begin
749           i := 1                                                      746
750         ; while ( s [ i ] <> '$' ) and ( i <= 100 )                   744   746   746
751           do begin
752             display ( s [ i ] )                                       141   744   746
753           ; i := i + 1                                                746   746
754           end
755         end { DisplayString }
756
757     ; procedure EmitError                                               *
758         ( errorCode : ErrorCodes ; sourceLineNo : integer )             *   289   383     S
759
760       ; begin
761           DisplayString ( 'epp $' )                                   744
762         ; case errorCode                                              758
763           of ePosStackOverflow                                        279
764             : DisplayString ( 'Position stack overflow $' )           744
765           ; ePosStackUnderflow                                        281
766             : DisplayString ( 'Position stack underflow $' )          744
767           ; eMarkStackOverflow                                        283
768             : DisplayString ( 'Mark stack overflow $' )               744
769           ; eMarkStackUnderflow                                       285
770             : DisplayString ( 'Mark stack underflow $' )              744
771           ; eCallStackOverflow                                        275
772             : DisplayString ( 'Call stack overflow $' )               744
773           end { case }
774         ; DisplayString ( 'at line $' )                               744
775         ; DisplayInteger ( sourceLineNo )                             725   758
776         ; Display ( nl )                                                *    32
777         ; processing := false                                         376     S
778         ; gError := errorCode                                         409   758
779         end { EmitError }
780
781     ; procedure getirec ( var p : linerecptr )                          *   170   329
782
783       ; begin
```

```
784           if gfreep = nil                                                    420
785           then begin new ( p ) ; linerecct := succ ( linerecct ) end          S   781   445     S   445
786           else begin p := gfreep ; gfreep := p ^ . dlink end                781   420   420   781   332
787         end " GETLREC "
788
789       ; procedure freelrec ( var p : linerecptr )                             *   781   329
790
791         ; begin
792             p ^ . dlink := gfreep                                           789   332   420
793           ; gfreep := p                                                     420   789
794           "% IF SINGLEPRECISION"
795           ; p ^ . tokno := 0                                                789   345
796           "% END SINGLEPRECISION"
797           "% IF MULTPRECISION"
798           ; multzero ( p ^ . tokno )                                        583   789   345
799           "% END MULTPRECISION"
800           end " FREELREC "
801
802       ; procedure unload                                                      *
803         ; forward                                                           *
804
805       ; procedure mark ( var m : markrec ; p : linerecptr )                   S     *   345   789   329
806
807         ; begin
808             with m do begin linep := p ; tokno := p ^ . tokno end           805   345   805   345   805   345
809           end " MARK "
810
811       ; procedure setupcmnt                                                   *
812
813         ; begin
814             with gcurp ^                                                    419
815             do begin
816               length := succ ( length )                                     336     S   336
817             ; aggsize := gaggsize                                           333   426
818             ; if symbol = tcomment2                                         439     *
819               then begin
820                 if startpos + length > rightmargin                          335   336   302
821                 then startpos := rightmargin - length                       335   302   336
822               ; if gcurpos < startpos                                       423   335
823                 then
824                 else if gcurpos + 1 + length <= rightmargin                 423   336   302
825                   then startpos := gcurpos + 1                              335   423
826                 "% IF SINGLEPRECISION"
827               ; gaggsize := gaggsize + length                               426   426   336
828                 "% END SINGLEPRECISION"
829                 "% IF MULTPRECISION"
830               ; multplint ( gaggsize , length , gaggsize )                  589   426   336   426
831                 "% END MULTPRECISION"
832               ; gcurpos := startpos + length                                423   335   336
833               ; newline := false                                            337     S
834               end
835               else if symbol = tcomment3                                    439     *
836                 then begin
837                   startpos := gcurpos                                       335   423
838                    "% IF SINGLEPRECISION"
839                 ; gaggsize := gaggsize + length                             426   426   336
```

```
840                "% END SINGLEPRECISION"                                                    *
841                "% IF MULTPRECISION"                                                       *
842              ; multplint ( gaggsize , length , gaggsize )                    589   426   336   426
843                "%  END MULTPRECISION"                                                    *
844              ; newline := false                                              337     S     *
845              ; gcurpos := startpos + length                                  423   335   336   *
846              end                                                             *     *     *     *
847              else if symbol = tcomment4                                      439     *     *     *
848                then begin                                                    *     *     *     *
849                    "% IF SINGLEPRECISION"                                    *     *     *     *
850                  gaggsize                                                    426   *     *     *
851                  := gaggsize                                                 426   *     *     *
852                     + length                                                 336   *     *     *
853                    "% END SINGLEPRECISION"                                   *     *     *     *
854                    "% IF MULTPRECISION"                                      *     *     *     *
855                       multplint ( gaggsize ,  length ,  gaggsize )          589   426   336   426
856                    "% END MULTPRECISION"                                     *     *     *     *
857                ; newline := true                                             337     S     *     *
858                ; gcurpos := startpos + length                                423   335   336   *
859                end                                                           *     *     *     *
860                else if symbol = tcomment1                                    439     *     *     *
861                  then begin                                                  *     *     *     *
862                    newline := true                                           337     S     *     *
863                  "% IF SINGLEPRECISION"                                      *     *     *     *
864                  ; gaggsize := gaggsize + rightmargin - leftmargin           426   426   302   301
865                  "% END SINGLEPRECISION"                                     *     *     *     *
866                  "% IF MULTPRECISION"                                        *     *     *     *
867                  ; multplint                                                 589   *     *     *
868                       ( gaggsize , rightmargin - leftmargin + 1              426   302   301   *
869                       , gaggsize )                                           426   *     *     *
870                  "% END MULTPRECISION"                                       *     *     *     *
871                  ; gcurpos := rightmargin                                    423   302   *     *
872                  end                                                         *     *     *     *
873          end                                                                 *     *     *     *
874        end " SETUPCMNT "                                                     *     *     *     *
875                                                                              *     *     *     *
876      ; procedure skipcmnt                                                    *     *     *     *
877                                                                              *     *     *     *
878        ; begin                                                               *     *     *     *
879            with gcurp ^                                                      419   *     *     *
880            do begin                                                          *     *     *     *
881              aggsize := gaggsize                                             333   426   *     *
882              "% IF SINGLEPRECISION"                                          *     *     *     *
883            ; gaggsize := gaggsize + length                                   426   426   336   *
884              "% END SINGLEPRECISION"                                         *     *     *     *
885              "% IF MULTPRECISION"                                            *     *     *     *
886            ; multplint ( gaggsize , length , gaggsize )                      589   426   336   426
887              "% END MULTPRECISION"                                           *     *     *     *
888            ; if symbol = tcomment3                                           439   835   *     *
889              then startpos := gcurpos + 1                                    335   423   *     *
890            ; if startpos + length > rightmargin                              335   336   302   *
891              then gcurpos := rightmargin                                     423   302   *     *
892              else gcurpos := startpos + length                               423   335   336   *
893            end                                                               *     *     *     *
894          end " SKIPCMNT "                                                    *     *     *     *
895                                                                              *     *     *     *
```

```
896     ; procedure incharacter                                          *
897
898       ; begin
899           if not eof ( input )                                       S     S
900           then begin
901             if eoln ( input )                                        S     S
902             then begin
903               pos := 1                                              412
904             ; sourceLineNo := succ ( sourceLineNo )                 758     S   758
905             ; innewline := true                                     413     S
906             end
907             else pos := succ ( pos )                                412     S   412
908           ; get ( input )                                           455     S
909           end
910         end " INCHARACTER "
911
912     ; procedure insertchar ( fc : char )                             *     *     S
913
914       ; var p : linerecptr                                          805   329
915
916       ; begin
917           if i > fraglength                                         746   307
918           then begin
919             getrec ( p )                                            781   914
920           ; p ^ . rlink := newp ^ . rlink                           914   332   417   332
921           ; newp ^ . rlink := p                                     417   332   914
922           ; newp := p                                               417   914
923           ; i := 1                                                  746
924           end
925         ; newp ^ . text [ i ] := fc                                 417     S   746   912
926         ; i := succ ( i )                                           746     S   746
927         ; newlength := succ ( newlength )                           411     S   411
928         end " INSERTCHAR "
929
930     ; procedure setupchar                                            *
931
932       ; begin
933           insertchar ( inputwindow ( input ) )                      912   571     S
934         ; incharacter                                               896
935         end " SETUPCHAR "
936
937     ; procedure prbracecmnt                                          *
938
939       ; begin
940           while ( not eoln ( input ) )                              S     S
941                 and ( inputwindow ( input ) <> '}' )                571     S
942           do setupchar                                              930
943         ; if eoln ( input )                                         S     S
944           then cmntstate := bracecomment                            415   415
945           else begin setupchar ; cmntstate := nocomment end         930   415   415
946         ; newsymbol := tcomment1                                    416   860
947         end " PRBRACECMNT "
948
949     ; function upperCase ( c : char ) : char                         *   527     S     S
950
951       ; begin
```

```
952           if ( ord ( 'a' "L.C." ) <= ord ( c ) )                                  S     S   949
953             and ( ord ( c ) <= ord ( 'z' "L.C." ) )                               S   949     S
954           then upperCase                                                          949
955             := chr ( ord ( c ) - ord ( 'a' "L.C." ) + ord ( 'A' ) )               S     S   949     S     S
956           else upperCase := c                                                     949   949
957         end
958
959     ; procedure lookupident                                                       *
960
961       ; var lstring : string12                                                    *   354
962         ; j , lo , hi , probe : integer                                           *     *     *     *     S
963         ; exitloop : boolean                                                      *     S
964
965       ; begin
966           if newlength > 12                                                       411
967           then newsymbol := tident                                                416     *
968           else begin
969             lstring := '            '                                             961
970           ; for j := 1 to newlength                                               962   411
971             do lstring [ j ] := upperCase ( newp ^ . text [ j ] )                 961   962   949   417     S   962
972           ; lo := rwptr [ newlength ] - 1                                         962   434   411
973           ; hi := rwptr [ newlength + 1 ]                                         962   434   411
974           ; exitloop := false                                                     963     S
975           ; repeat if lo + 1 >= hi                                                962   962
976               then begin newsymbol := tident ; exitloop := true end               416   967   963     S
977               else begin
978                 probe := ( lo + hi ) div 2                                        962   962   962
979               ; if rw [ probe ] . string = lstring                                435   962   438   961
980                 then begin
981                   newsymbol := rw [ probe ] . symbol                              416   435   962   439
982                 ; exitloop := true                                              963     S
983                 end
984                 else if rw [ probe ] . string < lstring                         435   962   438   961
985                   then lo := probe                                              962   962
986                   else hi := probe                                              962   962
987               end
988             until exitloop                                                      963
989           end
990         end " LOOKUPIDENT "
991
992     ; procedure number ( digits : setofchar )                                     *   442   328
993
994       ; begin
995           while inputwindow ( input ) in digits do setupchar                    571     S   992   930
996         ; if inputwindow ( input ) = '#'                                        571     S
997           then begin setupchar ; setupchar end                                  930   930
998         ; newsymbol := tnumber                                                  416     *
999         end " NUMBER "
1000
1001    "% IF PREP"
1002
1003    ; function prepcommand ( flp : linerecptr ) : boolean                        *     *   329     S
1004
1005      ; begin
1006          "ASSERT: FLP POINTS TO A COMMENT"
1007          with flp ^ . rlink ^                                                  1003   332
```

```
1008            do if text [ 1 ] = '{'                                          S
1009              then prepcommand := text [ 3 ] = '%'                        1003    S
1010              else prepcommand := text [ 2 ] = '%'                        1003    S
1011          end " PREPCOMMAND "
1012
1013      "% END PREP"
1014
1015      ; procedure processDollarSign                                         *
1016
1017        ; begin
1018            setupchar                                                     930
1019          ; if inputwindow ( input )                                      571     S
1020               in [ 'E' , 'F' , 'N' , 'T' , '***' , 'S' ]
1021            then setupchar                                                930
1022            else if inputwindow ( input ) in digits                       571     S     992
1023              then for i := 1 to 3 do setupchar                           746     930
1024          end " PROCESS_DOLLAR_SIGN "
1025
1026      ; procedure insymbol ( skipping : boolean )                           *       *     S
1027
1028        ; var isy : symboltyp                                               *     242
1029          ; laggsize : multint                                              *     325
1030          ; inextp , iip1 , iip2 : linerecptr                               *       *     *     329
1031          ; ic : char                                                       *       S
1032          ; exitloop : boolean                                            963       S
1033
1034        ; begin
1035            laggsize := gaggsize                                          1029    426
1036          ; if gcursy <> tendoffile                                       422       *
1037            then begin
1038              inextp := gnextp                                            1030    419
1039            ; repeat getirec ( newp )                                     781     417
1040              ; newp ^ . rlink := newp                                    417     332     417
1041              ; i := 1                                                    746
1042              ; newlength := 0                                            411
1043              ; if eoln ( input )                                           S       S
1044                then incharacter                                          896
1045              ; newstartpos := pos                                        412     412
1046              ; if cmntstate = bracecomment                               415     415
1047                then prbracecmnt                                          937
1048                else repeat exitloop := true                              1032    S
1049                  ; newstartpos := pos                                    412     412
1050                  ; if eof ( input )                                        S       S
1051                    then newsymbol := tendoffile                          416     1036
1052                    else if eoln ( input )                                  S       S
1053                      then begin incharacter ; exitloop := false end      896     1032    S
1054                      else if inputwindow ( input ) in lexchars           571     S     442
1055                        then case inputwindow ( input )                   571     S
1056                          of 'A' , 'B' , 'C' , 'D' , 'E' , 'F' , 'G'
1057                             , 'H' , 'I' , 'J' , 'K' , 'L' , 'M' , 'N'
1058                             , 'O' , 'P' , 'Q' , 'R' , 'S' , 'T' , 'U'
1059                             , 'V' , 'W' , 'X' , 'Y' , 'Z' , 'a' , 'b'
1060                             , 'c' , 'd' , 'e' , 'f' , 'g' , 'h' , 'i'
1061                             , 'j' , 'k' , 'l' , 'm' , 'n' , 'o' , 'p'
1062                             , 'q' , 'r' , 's' , 't' , 'u' , 'v' , 'w'
1063                             , 'x' , 'y' , 'z'
```

```
1064                    : begin
1065                      while inputwindow ( input )                    571     S
1066                            in letters + digits + [ '_' ]            442   992
1067                      do setupchar                                   930
1068                    ; lookupident                                    959
1069                    end
1070                  ; '''' 
1071                    : begin
1072                      setupchar                                      930
1073                    ; if not eof ( input )                             S     S
1074                         and not eoln ( input )                        S     S
1075                      then while ( inputwindow ( input ) <> ''''     571     S
1076                                 )
1077                                 and ( not eof ( input ) )             S     S
1078                                 and ( not eoln ( input ) )            S     S
1079                        do if inputwindow ( input ) = '$'            571     S
1080                          then processDollarSign                    1015
1081                          else setupchar                             930
1082                    ; if inputwindow ( input ) = ''''                571     S
1083                         then setupchar                              930
1084                    ; newsymbol := tstring                           416     *
1085                    end
1086                  ; '+'
1087                    : begin
1088                      setupchar                                      930
1089                    ; newsymbol := tplus                             416     *
1090                    end
1091                  ; '-'
1092                    : begin
1093                      setupchar                                      930
1094                    ; if inputwindow ( input ) = '>'                 571     S
1095                      then begin
1096                        setupchar                                    930
1097                      ; newsymbol := timplies                        416     *
1098                      end
1099                      else newsymbol := tminus                       416     *
1100                    end
1101                  ; '0' , '1' , '2' , '3' , '4' , '5' , '6'
1102                    , '7' , '8' , '9'
1103                    : number ( digits )                              992   992
1104                  ; '<'
1105                    : begin
1106                      setupchar                                      930
1107                    ; if inputwindow ( input ) = '='                 571     S
1108                      then begin
1109                        setupchar                                    930
1110                      ; newsymbol := tlessequal                      416     *
1111                      end
1112                      else newsymbol := tlessthan                    416     *
1113                    end
1114                  ; '>'
1115                    : begin
1116                      setupchar                                      930
1117                    ; if inputwindow ( input ) = '='                 571     S
1118                      then begin
1119                        setupchar                                    930
```

```
1120                        ; newsymbol := tgreaterequal                416      *
1121                        end
1122                        else newsymbol := tgreaterthan              416      *
1123                      end
1124                    ; '('
1125                      : begin
1126                        setupchar                                   930
1127                      ; newsymbol := tleftparen                     416      *
1128                      end
1129                    ; ':'
1130                      : begin
1131                        setupchar                                   930
1132                      ; if inputwindow ( input ) = '='              571      S
1133                        then begin
1134                          setupchar                                 930
1135                        ; newsymbol := tassign                      416      *
1136                        end
1137                        else newsymbol := tcolon                    416      *
1138                      end
1139                    ; '.'
1140                      : begin
1141                        if ellpsissw                                414
1142                        then begin
1143                          newstartpos := firstdotpos                412      412
1144                        ; insertchar ( '.' )                        912
1145                        ; setupchar                                 930
1146                        ; newsymbol := tdoubleperiod                416      *
1147                        ; ellpsissw := false                        414      S
1148                        end
1149                        else begin
1150                          setupchar                                 930
1151                        ; if inputwindow ( input ) = '.'            571      S
1152                          then begin
1153                            setupchar                               930
1154                          ; newsymbol := tdoubleperiod              416      1146
1155                          end
1156                          else newsymbol := tperiod                 416      *
1157                        end
1158                      end
1159                    ; '(' : begin setupchar ; prbracecmnt end       930      937
1160                    ; ')'
1161                      : begin
1162                        setupchar                                   930
1163                      ; newsymbol := trightparen                    416      *
1164                      end
1165                    ; '*'
1166                      : begin
1167                        setupchar                                   930
1168                      ; newsymbol := tasterisk                      416      *
1169                      end
1170                    ; '$'
1171                      : begin
1172                        setupchar                                   930
1173                      ; if inputwindow ( input ) = '$'              571      S
1174                        then processDollarSign                      1015
1175                        else setupchar                              930
```

```
1176                        ; newsymbol := tchar                          416      *
1177                        end
1178                      ; '~'
1179                        : begin
1180                          setupchar                                   930
1181                        ; if inputwindow ( input ) = '>'              571      S
1182                          then begin
1183                            setupchar                                 930
1184                          ; newsymbol := tlabel                       416      *
1185                          end
1186                          else newsymbol := tequal                    416      *
1187                        end
1188                      ; '/' , '[' , ']' , ',' , ';' , '&' , '@'
1189                      , '}'
1190                          "% IF ASCII"
1191                      , '^' "% END ASCII"
1192                        : begin
1193                          case inputwindow ( input )                  571      S
1194                          of '/' : isy := tunknown                   1028      *
1195                          ; '[' : isy := tunknown                    1028   1194
1196                          ; ']' : isy := tunknown                    1028   1194
1197                          ; ',' : isy := tcomma                      1028      *
1198                          ; '@'
1199                              "% IF ASCII"
1200                            , '^' "% END ASCII"
1201                            : isy := tuparrow                        1028      *
1202                          ; ';' : isy := tsemicolon                  1028      *
1203                          ; '&' : isy := tunknown                    1028   1194
1204                          ; '}' : isy := tunknown                    1028   1194
1205                          end
1206                        ; setupchar                                   930
1207                        ; newsymbol := isy                            416   1028
1208                        end
1209                      ; ' '
1210                        : begin
1211                          incharacter                                 896
1212                        ; exitloop := false                          1032      S
1213                        end
1214                      end " CASE "
1215                    else   " NOT INPUTWINDOW ( INPUT ) IN LEXCHARS "
1216                    begin
1217                      while ( not eoln ( input ) )                      S      S
1218                            and not ( inputwindow ( input )           571      S
1219                                      in lexchars                     442
1220                                    )
1221                      do setupchar                                    930
1222                    ; newsymbol := tunknown                           416   1194
1223                    end
1224                until exitloop                                       1032
1225              ; insertchar ( ' ' )                                    912
1226              ; with newp ^                                           417
1227                do begin
1228                  length := pred ( newlength )                        336      S    411
1229                ; symbol := newsymbol                                 439    416
1230                ; startpos := newstartpos                             335    412
1231                ; newline := innewline                                337    413
```

```
1232              ; dlink := nil                                                  332
1233              "% IF SINGLEPRECISION"
1234              ; gtokno := succ ( gtokno )                                     425      S    425
1235              "% END SINGLEPRECISION"
1236              "% IF MULTPRECISION"
1237              ; multplint ( gtokno , 1 , gtokno )                             589    425    425
1238              "% END MULTPRECISION"
1239              ; tokno := gtokno                                               345    425
1240              ; sepafter                                                      339
1241                := inputwindow ( input ) in [ ' ' ] + throwaways              571      S    443
1242              end
1243            ; inextp ^ . dlink := newp                                      1030    332    417
1244            ; inextp := newp                                                1030    417
1245            ; innewline := false                                             413      S
1246            until not ( newsymbol                                            416
1247                        in [ tcomment1 , tcomment2 , tcomment3               860    818    835
1248                           , tcomment4 ]                                     847
1249                      )
1250          ; tokenct := succ ( tokenct )                                      445      S    445
1251          ; gcompp := gcurp                                                  419    419
1252          ; while gcurp <> gnextp                                            419    419
1253            do begin
1254              if gcurp ^ . symbol                                            419    439
1255                 in [ tcomment1 , tcomment2 , tcomment3 , tcomment4 ]        860    818    835    847
1256              then if skipping then skipcmnt else setupcmnt                 1026    876    811
1257            ; gtrailp := gcurp                                               419    419
1258            ; gcurp := gcurp ^ . dlink                                       419    419    332
1259            ; if not ( gcompp ^ . symbol in [ tcomment1 , tcomment4 ]        419    439    860    847
1260                     )
1261              then begin gcomptrailp := gtrailp ; gcompp := gcurp end        419    419    419    419
1262            end
1263          ; gcurp ^ . aggsize := gaggsize                                    419    333    426
1264          ; gnextp := inextp                                                 419   1030
1265          ; gcursy := gcurp ^ . symbol                                       422    419    439
1266          ; llp1 := gcurp                                                   1030    419
1267          ; llp2 := llp1 ^ . dlink                                          1030   1030    332
1268          ; while llp2 <> gnextp                                            1030    419
1269            do begin
1270              with llp2 ^                                                   1030
1271              do begin
1272                if newline                                                   337
1273                then if dlink ^ . newline                                    332    337
1274                  then symbol := tcomment1                                   439    860
1275                  else symbol := tcomment4                                   439    847
1276                else if llp1 ^ . startpos + llp1 ^ . length + 1             1030    335   1030    336
1277                        >= startpos                                          335
1278                  then symbol := tcomment3                                   439    835
1279                  else symbol := tcomment2                                   439    818
1280              "% IF PREP"
1281              ; moveable := not prepcommand ( llp2 )                         340   1003   1030
1282              "% END PREP"
1283              "% IF NOT PREP"
1284              ; moveable := true                                             340      S
1285              "% END NOT PREP"
1286              end
1287            ; llp1 := llp2                                                  1030   1030
```

```
1288            ; llp2 := llp2 ^ . dlink                                  1030  1030   332
1289            end
1290          " ; IF GAGGSIZE - LAGGSIZE > RIGHTMARGIN - LEFTMARGIN
c 1291            THEN ** UNLOAD ** "
1292          end " IF GCURSY <> SYEOF "
1293        end " INSYMBOL "
1294
1295      ; procedure blankline                                             *
1296
1297        ; var p : linerecptr                                            914   329
1298          ; itokno , laggsize : multint                                   *  1029   325
1299
1300        ; begin
1301            getirec ( p )                                               781  1297
1302          "% IF SINGLEPRECISION"
1303          ; gtokno := succ ( gtokno )                                   425     S   425
1304          "% END SINGLEPRECISION"
1305          "% IF MULTPRECISION"
1306          ; multplint ( gtokno , 1 , gtokno )                           589   425   425
1307          "% END MULTPRECISION"
1308          ; itokno := gcurp ^ . tokno                                  1298   419   345
1309          ; with p ^                                                   1297
1310            do begin
1311              symbol := tblankline                                      439     *
1312            ; rlink := p                                                332  1297
1313            ; dlink := gcurp                                            332   419
1314            ; length := rightmargin - leftmargin + 1                    336   302   301
1315            end
1316          ; laggsize := gcurp ^ . aggsize                              1298   419   333
1317          ; if gfirstp = gcurp                                          419   419
1318            then gfirstp := p                                           419  1297
1319            else gtrailp ^ . dlink := p                                 419   332  1297
1320          ; gtrailp := p                                                419  1297
1321          "% IF SINGLEPRECISION"
1322          ; gaggsize := gaggsize + rightmargin - leftmargin + 1         426   426   302   301
1323          "% END SINGLEPRECISION"
1324          "% IF MULTPRECISION"
1325          ; multplint                                                   589
1326              ( gaggsize , rightmargin - leftmargin + 1 , gaggsize )    426   302   301   426
1327          "% END MULTPRECISION"
1328          ; if gcompp = gcurp                                           419   419
1329            then begin
1330              gcompp := p                                               419  1297
1331            ; gcomptrailp := nil " NOT THE RIGHT VALUE FOR              419
c 1332                                 GCOMPTRAILP, BUT IT WOULD
c 1333                                 TAKE A LOOP TO FIND, AND
c 1334                                 WE CANNOT USE IT ANYWAY "
1335            end
1336            else if gcompp ^ . symbol <> tblankline                     419   439  1311
1337              then begin
1338                getirec ( p )                                           781  1297
1339              "% IF SINGLEPRECISION"
1340              ; gtokno := succ ( gtokno )                               425     S   425
1341              "% END SINGLEPRECISION"
1342              "% IF MULTPRECISION"
1343              ; multplint ( gtokno , 1 , gtokno )                       589   425   425
```

```
1344            "% END MULTPRECISION"
1345            ; itokno := gcompp ^ . tokno                                    1298   419   345
1346            ; with p ^                                                      1297
1347              do begin
1348                symbol := tblankline                                          439  1311
1349              ; rlink := p                                                    332  1297
1350              ; dlink := gcompp                                               332   419
1351              ; length := rightmargin - leftmargin + 1                        336   302   301
1352              end
1353            ; laggsize := gcompp ^ . aggsize                                1298   419   333
1354            ; if gfirstp = gcompp                                            419   419
1355              then begin
1356                gfirstp := p                                                  419  1297
1357              ; gcomptrailp := p                                              419  1297
1358              ; gtrailp := p                                                  419  1297
1359              end
1360              else gcomptrailp ^ . dlink := p                                 419   332  1297
1361            "% IF SINGLEPRECISION"
1362            ; gaggsize := gaggsize + rightmargin - leftmargin + 1           426   426   302   301
1363            "% END SINGLEPRECISION"
1364            "% IF MULTPRECISION"
1365            ; multplint                                                      589
1366                ( gaggsize , rightmargin - leftmargin + 1 , gaggsize )      426   302   301   426
1367            "% END MULTPRECISION"
1368            ; gcompp := p                                                    419  1297
1369            end
1370          ; while p <> nil                                                  1297
1371            do with p ^                                                     1297
1372              do begin
1373                tokno := itokno                                              345  1298
1374              "% IF SINGLEPRECISION"
1375              ; itokno := succ ( itokno )                                   1298     S  1298
1376              "% END SINGLEPRECISION"
1377              "% IF MULTPRECISION"
1378              ; multplint ( itokno , 1 , itokno )                            589  1298  1298
1379              "% END MULTPRECISION"
1380              ; aggsize := laggsize                                          333  1298
1381                "% IF SINGLEPRECISION"
1382              ; laggsize := laggsize + length                               1298  1298   336
1383                "% END SINGLEPRECISION"
1384                "% IF MULTPRECISION"
1385              ; multplint ( laggsize , length , laggsize )                   589  1298   336  1298
1386                "% END MULTPRECISION"
1387              ; p := dlink                                                  1297   332
1388              end
1389          end " BLANKLINE "
1390
1391      ; procedure unload                                                     802
1392
1393        ; var lpos : positiontyp                                              *   320
1394          ; llp1 , llp2 , llp3 , llp4 : linerecptr                         1030  1030     *     *   329
1395          ; i : integer                                                     746     S
1396          "% IF MULTPRECISION"
1397          ; lmult : multint                                                   *   325
1398
1399          "% END MULTPRECISION"
```

```
1400
1401        ; begin
1402          "% IF MULTPRECISION"
1403            multsubtr                                                    618
1404              ( gcompp ^ . aggsize , gfirstp ^ . aggsize , imult )       419   333   419   333  1397
1405          ; if ( gfirstp <> gcompp )                                     419   419
1406               and ( multgt ( imult , linespace )                        681  1397   432
1407                     or ( gcompp ^ . symbol = tendoffile )               419   439  1036
1408                   )
1409            then
1410          "% END MULTPRECISION"
1411          "% IF SINGLEPRECISION"
1412              if ( gfirstp <> gcompp )                                   419   419
1413                 and ( ( gcompp ^ . aggsize - gfirstp ^ . aggsize        419   333   419   333
1414                         > rightmargin - leftmargin + 1                  302   301
1415                       )
1416                       or ( gcompp ^ . symbol = tendoffile )             419   439  1036
1417                     )
1418              then
1419          "% END SINGLEPRECISION"
1420              begin
1421                llp1 := gfirstp                                         1394   419
1422              ; while llp1 <> gcompp                                    1394   419
1423                do begin
1424                  if llp1 ^ . newline                                   1394   337
1425                     or ( llp1 ^ . startpos < outpos )                  1394   335   423
1426                     or ( llp1 ^ . symbol = tblankline )                1394   439  1311
1427                  then begin
1428                    writeln ( output )                                     S      S
1429                  ; outpos := 1                                          423
1430                  ; nltokno := llp1 ^ . tokno                            425  1394   345
1431                  ; lineoutct := succ ( lineoutct )                      445      S   445
1432                  end
1433                ; if llp1 ^ . symbol = tblankline                       1394   439  1311
1434                  then outpos := rightmargin                             423   302
1435                  else begin
1436                    while outpos < llp1 ^ . startpos                     423  1394   335
1437                    do begin
1438                      write ( output , ' ' )                                 S      S
1439                    ; outpos := succ ( outpos )                          423      S   423
1440                    end
1441                  ; llp2 := llp1 ^ . rlink                               1394  1394   332
1442                  ; i := 1                                               1395
1443                  ; for lpos := 1 to llp1 ^ . length                     1393  1394   336
1444                    do begin
1445                      if i > fraglength                                  1395   307
1446                      then begin llp2 := llp2 ^ . rlink ; i := 1 end     1394  1394   332  1395
1447                    ; write ( output , llp2 ^ . text [ i ] )                 S      S  1394     S  1395
1448                    ; i := succ ( i )                                    1395      S  1395
1449                    end
1450                  ; outpos := outpos + llp1 ^ . length                   423   423  1394   336
1451                  end
1452                ; outtokno := llp1 ^ . tokno                             425  1394   345
1453                ; llp2 := llp1                                           1394  1394
1454                ; llp1 := llp1 ^ . dlink                                 1394  1394   332
1455                ; llp4 := llp2                                           1394  1394
```

```
1456                ; repeat llp3 := llp2                                  1394  1394
1457                  ; llp2 := llp2 ^ . rlink                             1394  1394   332
1458                  ; freelrec ( llp3 )                                   789  1394
1459                  until llp2 = llp4                                    1394  1394
1460                end " WHILE "
1461              ; gfirstp := llp1                                        419  1394
1462              ; if gcompp = gcurp                                      419   419
1463                then gtrailp := nil                                    419
1464              ; gcomptrailp := nil                                     419
1465              end " IF GFIRSTP <> GCOMPP ETC. "
1466            end " UNLOAD "
1467
1468        ; procedure PushConstPos ( element : positiontyp )               *     *   320
1469
1470          ; begin
1471              if PosStackTop < MaxPosStack                             429   303
1472              then begin
1473                PosStackTop := PosStackTop + 1                         429   429
1474              ; PosStack [ PosStackTop ] := element                    427   429  1468
1475              end
1476              else begin
1477                EmitError ( ePosStackOverflow , sourceLineNo )         757   279   758
1478              ; processing := false                                    376     S
1479              end
1480            end  "  PushConstPos  "
1481
1482        ; procedure PushPos ( element : Positiontyp )                    *  1468     *
1483
1484          ; begin
1485              if PosStackTop < MaxPosStack                             429   303
1486              then begin
1487                PosStackTop := PosStackTop + 1                         429   429
1488              ; PosStack [ PosStackTop ] := element                    427   429  1482
1489              end
1490              else begin
1491                EmitError ( ePosStackOverflow , sourceLineNo )         757   279   758
1492              ; processing := false                                    376     S
1493              end
1494            end  "  PushPos  "
1495
1496        ; procedure PushMark ( m : markrec )                             *   805   345
1497
1498          ; begin
1499              if MarkStackTop < MaxMarkStack                           430   304
1500              then begin
1501                MarkStackTop := MarkStackTop + 1                       430   430
1502              ; MarkStack [ MarkStackTop ] := m                        428   430  1496
1503              end
1504              else begin
1505                EmitError ( eMarkStackOverflow , sourceLineNo )        757   283   758
1506              ; processing := false                                    376     S
1507              end
1508            end  "  PushMark  "
1509
1510        ; procedure PopPos                                               *
1511
```

```
1512        ; begin
1513            if PosStackTop > 0                                                  429
1514            then PosStackTop := PosStackTop - 1                                 429    429
1515            else begin
1516              EmitError ( ePosStackUnderflow , sourceLineNo )                   757    281    758
1517            ; processing := false                                               376      S
1518            end
1519          end  "  PopPos  "
1520
1521      ; procedure PopMark                                                         *
1522
1523        ; begin
1524            if MarkStackTop > 0                                                 430
1525            then MarkStackTop := MarkStackTop - 1                               430    430
1526            else begin
1527              EmitError ( eMarkStackUnderflow , sourceLineNo )                  757    285    758
1528            ; processing := false                                               376      S
1529            end
1530          end  "  PopMark  "
1531
1532      ; procedure setup ( fpos : positiontyp ; fnewline : boolean )             *      *    320      *      S
1533
1534        ; var llength , lstart : integer                                        *      *      S
1535
1536        ; begin
1537            with gcurp ^                                                        419
1538            do begin
1539              length := succ ( length )                                         336      S    336
1540            ; if dlink ^ . symbol = tcomment3                                   332    439    835
1541              then llength := length + dlink ^ . length                        1534    336    332    336
1542              else llength := length                                           1534    336
1543            ; if fpos + llength > rightmargin                                  1532   1534    302
1544              then begin
1545                lstart := rightmargin - llength                                1534    302   1534
1546              ; if lstart < leftmargin                                         1534    301
1547                then startpos := leftmargin                                    335    301
1548                else startpos := lstart                                        335   1534
1549              end
1550              else startpos := fpos                                            335   1532
1551            ; newline := fnewline                                              337   1532
1552            ; aggsize := gaggsize                                              333    426
1553            "% IF SINGLEPRECISION"
1554            ; gaggsize := gaggsize + length                                    426    426    336
1555            "% END SINGLEPRECISION"
1556            "% IF MULTPRECISION"
1557            ; multplint ( gaggsize , length , gaggsize )                       589    426    336    426
1558            "% END MULTPRECISION"
1559              ; gcurpos := startpos + length                                   423    335    336
1560          " ASSERT: NOT ( SYMBOL
c 1561                        IN [ TCOMMENT1 , TCOMMENT2 , TCOMMENT3 , TCOMMENT4 ]
c 1562                        ) "
1563            ; gcurposnc := gcurpos                                             423    423
1564            ; skip := false                                                    338      S
1565            ; insympool ( false )                                             1026      S
1566            end
1567          end " SETUP "
```

```
1568
1569      ; procedure ccompress ( m : markrec )                                  *  1496   345
1570
1571        ; var lpos : positiontyp                                          1393   320
1572          ; llp : linerecptr                                                *   329
1573          "% IF MULTPRECISION"
1574          ; lmult : multint                                               1397   325
1575
1576          "% END MULTPRECISION"
1577
1578        ; begin
1579          "% IF MULTPRECISION"
1580            multplint ( m . tokno , 1 , lmult )                            589  1569   345  1574
1581          ; if multgt ( gcompp ^ . tokno , lmult )                         681   419   345  1574
1582               and multge ( m . tokno , nltokno )                          672  1569   345   425
1583            then
1584          "% END MULTPRECISION"
1585          "% IF SINGLEPRECISION"
1586              if ( m . tokno + 1 < gcompp ^ . tokno )                      1569   345   419   345
1587                and ( m . tokno >= nltokno )                               1569   345   425
1588              then
1589          "% END SINGLEPRECISION"
1590              begin
1591            "% IF SINGLEPRECISION"
1592                if m . tokno > outtokno                                    1569   345   425
1593                then
1594            "% END SINGLEPRECISION"
1595            "% IF MULTPRECISION"
1596                  if multgt ( m . tokno , outtokno )                        681  1569   345   425
1597                  then
1598            "% END MULTPRECISION"
1599                  begin                                                   1571
1600                    lpos                                                  1569   345   335  1569   345   336
1601                    := m . linep ^ . startpos + m . linep ^ . length      1572  1569   345   332
1602                  ; llp := m . linep ^ . dlink
1603                  end
1604                  else begin llp := gfirstp ; lpos := outpos end          1572   419  1571   423
1605          "% IF MULTPRECISION"
1606              ; multsubtr                                                  618
1607                  ( gcompp ^ . aggsize , llp ^ . aggsize , lmult )         419   333  1572   333  1574
1608              ; multplint ( lmult , lpos , lmult )                         589  1574  1571  1574
1609              ; if multge ( multrightmargin , lmult )                      672   432  1574
1610                then if multgt ( gcompp ^ . tokno , llp ^ . tokno ) then   681   419   345  1572   345
1611          "% END MULTPRECISION"
1612              "% IF SINGLEPRECISION"
1613              ; if ( lpos + gcompp ^ . aggsize - llp ^ . aggsize )         1571   419   333  1572   333
1614                   <= rightmargin                                          302
1615                then if llp ^ . tokno < gcompp ^ . tokno                  1572   345   419   345
1616                  then
1617              "% END SINGLEPRECISION"
1618                  begin
1619                    while llp <> gcompp                                   1572   419
1620                    do begin
1621                      if llp ^ . symbol in [ tcomment2 , tcomment4 ]      1572   439   818   847
1622                      then lpos := llp ^ . dlink ^ . startpos             1571  1572   332   335
1623                      else begin
```

```
1624                      llp ^ . startpos := lpos                          1572   335  1571
1625                    ; lpos := lpos + llp ^ . length                     1571  1571  1572   336
1626                    ; if llp ^ . skip                                   1572   338
1627                      then lpos := succ ( lpos )                        1571     S  1571
1628                    ; llp ^ . newline := false                          1572   337     S
1629                    end
1630                  ; llp := llp ^ . dlink                                1572  1572   332
1631                  end
1632                ; if gcompp = gcurp                                      419   419
1633                  then gcurpos := lpos                                   423  1571
1634                ; gcurposnc := lpos                                      423  1571
1635                end
1636                else
1637              else unload                                               1391
1638            end
1639        end " CCOMPRESS "
1640
1641    ; procedure movecmnts                                                 *
1642
1643      ; var lcurp : linerecptr                                            *   329
1644        ; laggsize , ltokno : multint                                   1298  1298   325
1645      "% IF MULTPRECISION"
1646        ; lmult : multint                                               1574   325
1647
1648      "% END MULTPRECISION"
1649
1650      ; begin
1651          laggsize := gaggsize                                          1644   426
1652        ; lcurp := gcurp                                                1643   419
1653        ; if ( gcurp ^ . dlink <> gnextp )                               419   332   419
1654             and ( gcurp ^ . dlink ^ . moveable )                        419   332   340
1655          then begin
1656            gcurp := gcurp ^ . dlink                                     419   419   332
1657          ; if gfirstp = lcurp                                           419  1643
1658            then gfirstp := gcurp                                        419   419
1659            else gtrailp ^ . dlink := gcurp                              419   332   419
1660          ; if gcompp = lcurp                                            419  1643
1661            then gcompp := gcurp                                         419   419
1662          ; ltokno := lcurp ^ . tokno                                   1644  1643   345
1663          ; while ( gcurp <> gnextp ) and ( gcurp ^ . moveable )          419   419   419   340
1664            do begin
1665              gcurp ^ . tokno := ltokno                                  419   345  1644
1666          "% IF SINGLEPRECISION"
1667            ; ltokno := succ ( ltokno )                                 1644     S  1644
1668          "% END SINGLEPRECISION"
1669          "% IF MULTPRECISION"
1670            ; multplint ( ltokno , 1 , ltokno )                          589  1644  1644
1671          "% END MULTPRECISION"
1672            ; setupcmnt                                                 811
1673            ; gtrailp := gcurp                                          419   419
1674            ; gcurp := gcurp ^ . dlink                                  419   419   332
1675            ; if not ( gcompp ^ . symbol in [ tcomment1 , tcomment4 ]   419   439   860   847
1676                     )
1677              then begin gcomptrailp := gtrailp ; gcompp := gcurp end    419   419   419   419
1678            end
1679          ; lcurp ^ . tokno := ltokno                                   1643   345  1644
```

```
1680          ; lcurp ^ . dlink := gcurp                                  1643   332   419
1681          ; gtrailp ^ . dlink := lcurp                                419   332  1643
1682          ; if gcompp = gcurp                                         419   419
1683            then gcompp := lcurp                                      419  1643
1684          ; gcurp := lcurp                                            419  1643
1685          ; gcurp ^ . aggsize := gaggsize                             419   333   426
1686        "% IF SINGLEPRECISION"
1687          ; if gaggsize - laggsize > rightmargin - leftmargin         426  1644   302   301
1688            then unload                                              1391
1689        "% END SINGLEPRECISION"
1690        "% IF MULTPRECISION"
1691          ; multsubtr ( gaggsize , laggsize , lmult )                 618   426  1644  1646
1692          ; if multge ( lmult , linespace )                           672  1646   432
1693            then unload                                              1391
1694        "% END MULTPRECISION"
1695          end
1696        end " MOVECMNTS "
1697
1698    ; procedure skip ( fsys : setofsymbol )                           338     *   243
1699
1700      ; var laggsize : multint                                       1644   325
1701        ; i : integer                                                1395     5
1702      "% IF MULTPRECISION"
1703        ; lmult : multint                                            1646   325
1704
1705      "% END MULTPRECISION"
1706
1707      ; begin
1708          laggsize := gaggsize                                       1700   426
1709        ; if displaysyntaxErrors                                       *
1710          then begin
1711          ; DisplayString ( 'skipSet$' )                             744
1712          ; for i := 0 to 127                                       1701
1713            do if i in fsys                                         1701  1698
1714              then DisplayInteger ( i )                              725  1701
1715          ; display ( nl )                                           141    32
1716          end
1717        ; while not ( gcursy in fsys )                               422  1698
1718          do begin
1719            with gcurp ^                                             419
1720            do begin
1721              skip := true                                          1698     5
1722            ; if displaySyntaxErrors                                 379
1723              then begin
1724              ; DisplayString ( 'skipSymbol$' )                      744
1725              ; DisplayInteger ( gcursy )                            725   422
1726              ; DisplayInteger ( tokno )                             725   345
1727              ; display ( nl )                                       141    32
1728              end
1729            ; aggsize := gaggsize                                    333   426
1730            ; if ( dlink ^ . symbol in fsys ) and sepafter           332   439  1698   339
1731              then length := succ ( length )                         336     5   336
1732          "% IF SINGLEPRECISION"
1733            ; gaggsize := gaggsize + length                          426   426   336
1734          "% END SINGLEPRECISION"
1735          "% IF MULTPRECISION"
```

```
1736            ; multplint ( gaggsize , length , gaggsize )                589   426   336   426
1737          "% END MULTPRECISION"
1738            ; skipct := succ ( skipct )                                 445     S   445
1739            ; if startpos + length > rightmargin                        335   336   302
1740              then gcurpos := rightmargin                               423   302
1741              else gcurpos := startpos + length                         423   335   336
1742            ; gcurposnc := gcurpos                                      423   423
1743          "% IF SINGLEPRECISION"
1744            ; if gaggsize - laggsize > rightmargin - leftmargin + 1     426  1700   302   301
1745              then begin unload ; laggsize := gaggsize end             1391  1700   426
1746          "% END SINGLEPRECISION"
1747          "% IF MULTPRECISION"
1748            ; multsubtr ( gaggsize , laggsize , lmult )                 618   426  1700  1703
1749            ; if multgt ( lmult , linespace )                           681  1703   432
1750              then begin unload ; laggsize := gaggsize end             1391  1700   426
1751          "% END MULTPRECISION"
1752            end
1753          ; insymbol ( true )                                          1026     S
1754          end
1755        ; if displaySyntaxErrors                                        379
1756          then begin
1757            DisplayString ( 'ResumeLine$' )                             744
1758          ; DisplayInteger ( sourceLineNo )                             725   758
1759          ; DisplayString ( 'token$' )                                  744
1760          ; DisplayInteger ( gcursy )                                   725   422
1761          ; DisplayInteger ( gcurp ^ . tokno )                          725   419   345
1762          ; display ( nl )                                              141    32
1763          end
1764        end " SKIP "
1765
1766    ; function indent ( fpos : positiontyp ) : positiontyp               *  1532   320   320
1767
1768      ; var i : integer                                                1701     S
1769
1770      ; begin
1771          i := fpos + indentamt                                        1768  1766   306
1772        ; if i <= rightmargin                                          1768   302
1773          then indent := i                                             1766  1768
1774          else indent := rightmargin                                   1766   302
1775        end " INDENT "
1776
1777    ; procedure inropenlist                                             *
1778        ( fpos : positiontyp                                          1766   320
1779        ; startmark : markrec                                            *   345
1780        ; syselembeg , sepsys : setofsymbol                              *     *   243
1781        ; elementhandler : parproc                                       *   346
1782        ; partcompress , flmovecmnts , indentf : boolean                 *     *     *     S
1783        ; stopsys : setofsymbol                                          *   243
1784        ; var returnmark : markrec                                       *   345
1785        )
1786
1787      ; var mark1 , mark2 : markrec                                      *     *   345
1788        ; lpos1 , lpos2 : positiontyp                                    *     *   320
1789        ; lsys , sysloop : setofsymbol                                   *     *   243
1790
1791      ; begin
```

```
1792          sysloop := syselembeg + sepsys                                  1789  1780  1780
1793        ; lsys := sysloop + stopsys                                       1789  1789  1783
1794        ; skip ( lsys )                                                   1698  1789
1795        ; mark ( mark1 , gcurp )                                             S  1787   419
1796        ; if indentf                                                      1782
1797          then lpos1 := indent ( fpos )                                   1788  1766  1778
1798          else lpos1 := fpos                                              1788  1778
1799        ; if ( gcursy in sepsys ) and flmovecmnts                          422  1780  1782
1800          then movecmnts                                                  1641
1801        ; while gcursy in sysloop                                          422  1789
1802          do begin
1803            mark ( mark2 , gcurp )                                           S  1787   419
1804          ; if gcursy in sepsys                                            422  1780
1805            then begin setup ( fpos , true ) ; lpos2 := gcurposnc end      1532  1778     S  1788   423
1806            else lpos2 := lpos1                                            1788  1788
1807          ; if ( gcursy in sepsys ) and flmovecmnts                        422  1780  1782
1808            then movecmnts                                                1641
1809          ; if partcompress                                               1782
1810            then ccompress ( mark1 )                                      1569  1787
1811          ; mark1 := mark2                                                1787  1787
1812          end
1813        ; returnmark := mark1                                             1784  1787
1814        end " INROPENLIST "
1815
1816      ; procedure openlist                                                    *
1817          ( fpos : positiontyp                                            1778   320
1818          ; startmark : markrec                                           1779   345
1819          ; syselembeg , sepsys : setofsymbol                             1780  1780   243
1820          ; elementhandler : parproc                                      1781   346
1821          ; partcompress , flmovecmnts , indentf : boolean                1782  1782  1782     S
1822          ; stopsys : setofsymbol                                         1783   243
1823          )
1824
1825        ; var mark : markrec                                                 S   345
1826
1827        ; begin
1828            inropenlist                                                   1777
1829              ( fpos , startmark , syselembeg , sepsys , elementhandler   1817  1818  1819  1819  1820
1830              , partcompress , flmovecmnts , indentf , stopsys , mark )   1821  1821  1821  1822     S
1831          end " OPENLIST "
1832
1833      ; procedure closedlist                                                  *
1834          ( fpos : positiontyp                                            1817   320
1835          ; startmark : markrec                                           1818   345
1836          ; sysopen , syselembeg , sepsys , sysclose : setofsymbol            *  1819  1819     *   243
1837          ; elementhandler : parproc                                      1820   346
1838          ; partcompress , flmovecmnts : boolean                          1821  1821     S
1839          ; stopsys : setofsymbol                                         1822   243
1840          )
1841
1842        ; var mark : markrec                                                 S   345
1843
1844        ; begin
1845            skip ( sysopen + syselembeg + sepsys + sysclose + stopsys )   1698  1836  1836  1836  1836  1839
1846          ; if gcursy in sysopen                                           422  1836
1847            then begin
```

```
1848            setup ( fpos , false )                                  1532  1834     S
1849          ; ccompress ( startmark )                                 1569  1835
1850          end
1851        ; inropenlist                                               1777
1852          ( fpos , startmark , syselembeg , sepsys , elementhandler 1834  1835  1836  1836  1837
1853          , partcompress , flmovecmnts , true , sysclose + stopsys  1838  1838     S  1836  1839
1854          , mark )                                                     S
1855        ; if gcursy in sysclose                                      422  1836
1856          then setup ( fpos , true )                                1532  1834     S
1857        ; if partcompress                                           1838
1858          then ccompress ( mark )                                   1569     S
1859        end " CLOSEDLIST "
1860
1861    ; procedure initrw ( fsymbol : symboltyp ; fstring : string12 )       *     *   242   693   354
1862
1863      ; var len , i , j : integer                                         *  1768   962     S
1864
1865      ; begin
1866          if rwct < rwmax - 1                                         441   308
1867          then begin
1868            len := 12                                                1863
1869          ; while ( len >= 2 ) and ( fstring [ len ] = ' ' )         1863  1861  1863
1870            do len := pred ( len )                                   1863     S  1863
1871          ; i := rwptr [ len ]                                       1863   434  1863
1872          ; j := rwptr [ len + 1 ]                                   1863   434  1863
1873          ; while ( i < j ) and ( rw [ i ] . string < fstring )      1863  1863   435  1863   438  1861
1874            do i := succ ( i )                                       1863     S  1863
1875          ; if rw [ i ] . string <> fstring                           435  1863   438  1861
1876            then begin
1877              j := rwptr [ 13 ] - 1                                  1863   434
1878            ; while j >= i                                           1863  1863
1879              do begin
1880                rw [ j + 1 ] := rw [ j ]                              435  1863   435  1863
1881              ; j := pred ( j )                                      1863     S  1863
1882              end
1883            ; rw [ i ] . string := fstring                           435  1863   438  1861
1884            ; for j := len + 1 to 13                                1863  1863
1885              do rwptr [ j ] := succ ( rwptr [ j ] )                 434  1863     S   434  1863
1886            end
1887          ; rw [ i ] . symbol := fsymbol                             435  1863   439  1861
1888          ; rwct := succ ( rwct )                                    441     S   441
1889          end
1890        end " INITRW "
1891
1892    ; procedure initdumirec ( var irp : linerecptr )                      *     *   329
1893
1894      ; begin
1895          with irp ^                                                 1892
1896          do begin
1897            dlink := nil                                              332
1898          ; rlink := irp                                              332  1892
1899          ; newline := false                                          337     S
1900        "% IF SINGLEPRECISION"
1901          ; tokno := 0                                                345
1902        "% END SINGLEPRECISION"
1903        "% IF MULTPRECISION"
```

```
1904          ; multzero ( tokno )                                          583   345
1905        "% END MULTPRECISION"
1906          ; symbol := tdummy                                            439    *
1907          ; startpos := leftmargin                                      335   301
1908          ; length := 0                                                 336
1909        "% IF SINGLEPRECISION"
1910          ; aggsize := 0                                                333
1911        "% END SINGLEPRECISION"
1912        "% IF MULTPRECISION"
1913          ; multzero ( aggsize )                                        583   333
1914        "% END MULTPRECISION"
1915          end
1916        end " INITDUMLREC "
1917
1918      ; procedure initsets                                               *
1919
1920        ; begin
1921            letters                                                     442
1922            := [ 'A' , 'B' , 'C' , 'D' , 'E' , 'F' , 'G' , 'H' , 'I'
1923               , 'J' , 'K' , 'L' , 'M' , 'N' , 'O' , 'P' , 'Q' , 'R'
1924               , 'S' , 'T' , 'U' , 'V' , 'W' , 'X' , 'Y' , 'Z' , 'a'
1925               , 'b' , 'c' , 'd' , 'e' , 'f' , 'g' , 'h' , 'i' , 'j'
1926               , 'k' , 'l' , 'm' , 'n' , 'o' , 'p' , 'q' , 'r' , 's'
1927               , 't' , 'u' , 'v' , 'w' , 'x' , 'y' , 'z' ]
1928          ; octaldigits                                                 442
1929            := [ '0' , '1' , '2' , '3' , '4' , '5' , '6' , '7' ]
1930          ; digits := octaldigits + [ '8' , '9' ]                       992   442
1931          ; hexdigits                                                   442
1932            := digits                                                   992
1933               + [ 'A' , 'B' , 'C' , 'D' , 'E' , 'F' , 'a' , 'b' , 'c'
1934                 , 'd' , 'e' , 'f' ]
1935          ; lexchars                                                    442
1936            := letters                                                  442
1937               + digits                                                 992
1938               + [ '"' , '$' , '+' , '-' , '>' , '<' , '(' , ':' , '.'
1939                 , ')' , '=' , '/' , '*' , '[' , ']' , ',' , '@' , ';'
1940                 , '&' , ' ' , '(' , ')' , '_'
1941                 "% IF ASCII"
1942                 , '^' "% END ASCII" ]
1943          ; throwaways := [ chr ( ht ) , chr ( ff ) ]                   443     S   314     S   315
1944          end " INITSETS "
1945
1946      ; procedure initrwtable                                            *
1947
1948        ; begin
1949            rwct := 0                                                   441
1950          ; for i := 1 to 13 do rwptr [ i ] := 1                        1863   434  1863
1951          ; initrw ( tabstraction , 'ABSTRACTION ' )                    1861    *
1952          ; initrw ( taligned , 'ALIGNED     ' )                        1861    *
1953          ; initrw ( tall , 'ALL         ' )                            1861    *
1954          ; initrw ( tand , 'AND         ' )                            1861    *
1955          ; initrw ( tany , 'ANY         ' )                            1861    *
1956          ; initrw ( tarray , 'ARRAY       ' )                          1861    *
1957          ; initrw ( tassert , 'ASSERT      ' )                         1861    *
1958          ; initrw ( tat , 'AT          ' )                             1861    *
1959          ; initrw ( tbegin , 'BEGIN       ' )                          1861    *
```

```
1960        ; initrw ( tbind , 'BIND          ' )                    1861    *
1961        ; initrw ( tbits , 'BITS          ' )                    1861    *
1962        ; initrw ( tbound , 'BOUND         ' )                   1861    *
1963        ; initrw ( tcase , 'CASE          ' )                    1861    *
1964        ; initrw ( tcheckable , 'CHECKABLE    ' )                1861    *
1965        ; initrw ( tchecked , 'CHECKED      ' )                  1861    *
1966        ; initrw ( tcode , 'CODE          ' )                    1861    *
1967        ; initrw ( tcollection , 'COLLECTION  ' )                1861    *
1968        ; initrw ( tconst , 'CONST        ' )                    1861    *
1969        ; initrw ( tconverter , 'CONVERTER    ' )                1861    *
1970        ; initrw ( tcounted , 'COUNTED      ' )                  1861    *
1971        ; initrw ( tdecreasing , 'DECREASING  ' )                1861    *
1972        ; initrw ( tdefault , 'DEFAULT      ' )                  1861    *
1973        ; initrw ( tdependent , 'DEPENDENT    ' )                1861    *
1974        ; initrw ( tdiv , 'DIV           ' )                     1861    *
1975        ; initrw ( telse , 'ELSE          ' )                    1861    *
1976        ; initrw ( telseif , 'ELSEIF        ' )                  1861    *
1977        ; initrw ( tend , 'END           ' )                     1861    *
1978        ; initrw ( texit , 'EXIT          ' )                    1861    *
1979        ; initrw ( texports , 'EXPORTS      ' )                  1861    *
1980        ; initrw ( texternal , 'EXTERNAL     ' )                 1861    *
1981        ; initrw ( tfinally , 'FINALLY      ' )                  1861    *
1982        ; initrw ( tfor , 'FOR           ' )                     1861    *
1983        ; initrw ( tforward , 'FORWARD      ' )                  1861    *
1984        ; initrw ( tfrom , 'FROM          ' )                    1861    *
1985        ; initrw ( tfunction , 'FUNCTION     ' )                 1861    *
1986        ; initrw ( tif , 'IF            ' )                      1861    *
1987        ; initrw ( timports , 'IMPORTS      ' )                  1861    *
1988        ; initrw ( tin , 'IN            ' )                      1861    *
1989        ; initrw ( tinclude , 'INCLUDE      ' )                  1861    *
1990        ; initrw ( tinitially , 'INITIALLY    ' )                1861    *
1991        ; initrw ( tinline , 'INLINE       ' )                   1861    *
1992        ; initrw ( tinvariant , 'INVARIANT    ' )                1861    *
1993        ; initrw ( tloop , 'LOOP          ' )                    1861    *
1994        ; initrw ( tmachine , 'MACHINE      ' )                  1861    *
1995        ; initrw ( tmod , 'MOD           ' )                     1861    *
1996        ; initrw ( tmodule , 'MODULE       ' )                   1861    *
1997        ; initrw ( tnot , 'NOT           ' )                     1861    *
1998        ; initrw ( tof , 'OF            ' )                      1861    *
1999        ; initrw ( tor , 'OR            ' )                      1861    *
2000        ; initrw ( totherwise , 'OTHERWISE    ' )                1861    *
2001        ; initrw ( tpacked , 'PACKED       ' )                   1861    *
2002        ; initrw ( tparameter , 'PARAMETER    ' )                1861    *
2003        ; initrw ( tpervasive , 'PERVASIVE    ' )                1861    *
2004        ; initrw ( tpost , 'POST          ' )                    1861    *
2005        ; initrw ( tpre , 'PRE           ' )                     1861    *
2006        ; initrw ( tprocedure , 'PROCEDURE    ' )                1861    *
2007        ; initrw ( treadonly , 'READONLY     ' )                 1861    *
2008        ; initrw ( trecord , 'RECORD       ' )                   1861    *
2009        ; initrw ( treturn , 'RETURN       ' )                   1861    *
2010        ; initrw ( treturns , 'RETURNS      ' )                  1861    *
2011        ; initrw ( tset , 'SET           ' )                     1861    *
2012        ; initrw ( tthen , 'THEN          ' )                    1861    *
2013        ; initrw ( tthus , 'THUS          ' )                    1861    *
2014        ; initrw ( tto , 'TO            ' )                      1861    *
2015        ; initrw ( ttype , 'TYPE          ' )                    1861    *
```

```
2016        ; initrw ( tunknown , 'UNKNOWN      ' )                          1861  1194
2017        ; initrw ( tvar , 'VAR          ' )                              1861    *
2018        ; initrw ( twhen , 'WHEN         ' )                             1861    *
2019        ; initrw ( twith , 'WITH         ' )                             1861    *
2020        ; initrw ( txor , 'XOR          ' )                              1861    *
2021        end " INITRWTABLE "
2022
2023      ; procedure initinp                                                   *
2024
2025        ; var istring : string12                                          961   354
2026          ; i , j : integer                                              1863  1863     S
2027
2028        ; begin
2029          "% IF PASCAL32"
2030            open ( initfile , 3 , opforin )                              489   450   360
2031          "% END PASCAL32"
2032          "% IF PASCAL"                       reset ( initfile )          S   450
2033          "% END PASCAL"
2034          ; get ( initfile )                                             455   450
2035          ; while not eof ( initfile )                                    S   450
2036            do begin
2037              istring := '            '                                  2025
2038            ; i := 1                                                     2026
2039            ; while not eof ( initfile )                                  S   450
2040                    and ( eoln ( initfile )                               S   450
2041                          or not ( inputwindow ( initfile ) in letters )  571   450   442
2042                        )
2043              do get ( initfile ) " SCAN FOR LETTER "                     455   450
2044          " ASSERT EOF OR (NOT EOLN AND ( INPUTWINDOW ( INITFILE ) IN LETTERS)) "
2045            ; if not eof ( initfile )                                     S   450
2046              then begin
2047                while not eof ( initfile )                                S   450
2048                      and not eoln ( initfile )                           S   450
2049                      and ( inputwindow ( initfile ) in letters )         571   450   442
2050                      and ( i <= 12 )                                    2026
2051                do begin
2052                  istring [ i ] := inputwindow ( initfile )              2025  2026   571   450
2053                ; i := succ ( i )                                        2026     S  2026
2054                ; get ( initfile )                                       455   450
2055                end
2056              ; while not eof ( initfile )                                S   450
2057                      and not eoln ( initfile )                           S   450
2058                      and ( inputwindow ( initfile ) in letters )         571   450   442
2059                do get ( initfile )                                      455   450
2060              " ASSERT: 1 <= I-1 = NO OF LETTERS <= 12 "
2061              ; for j := rwptr [ i - 1 ] to rwptr [ i ] - 1              2026   434  2026   434  2026
2062                do with rw [ j ]                                         435  2026
2063                  do if string = istring                                 438  2025
2064                    then symbol := tident                                439   967
2065              end
2066            end
2067          "% IF PASCAL32"
2068          ; close ( initfile )                                           553   450
2069          "% END PASCAL32"
2070          end {SSL Interpreter, written in Pascal.}
2071
```

```
2072 ( Jan 5, 1983 )
2073
2074     ; procedure SyntaxError ( sourceLineNo , SSLLineNo : integer )          *   758      *     S
2075
2076       ; begin
2077           if displaySyntaxErrors                                           379
2078           then begin
2079             DisplayString ( 'SyntaxErr, srcLine$' )                        744
2080           ; DisplayInteger ( sourceLineNo )                                725  2074
2081           ; DisplayString ( 'S/SLLine$' )                                  744
2082           ; DisplayInteger ( SSLLineNo )                                   725  2074
2083           ; DisplayString ( 'token$' )                                     744
2084           ; DisplayInteger ( gcursy )                                      725  422
2085           ; DisplayInteger ( gcurp ^ . tokno )                             725  419   345
2086           ; display ( nl )                                                 141   32
2087           end
2088         end
2089
2090     ; procedure Check ( SSLLineNo : integer ; keys : setofsymbol )          *   2074     S    384   243
2091
2092             { This procedure checks if the incoming token is in the
c 2093            current recovery token set and emits an error message if not }
2094
2095       ; begin
2096           if not ( gcursy in keys )                                        422  2090
2097           then begin
2098             SyntaxError ( sourceLineNo , SSLLineNo )                       2074 2074  2090
2099           ; skip ( keys )                                                  1698 2090
2100           end
2101         end {Check}
2102
2103     ; procedure ConvertToKeySet                                            *
2104         ( address : integer ; var result : setofsymbol )                   S     S    619   243
2105
2106       ; var irec                                                           *
2107           : record
2108               case boolean                                                 S
2109               of false : ( ary : setArray )                                S     *    *
2110               ; true : ( pset : setofsymbol )                              S     *    243
2111             end
2112       ; i : 1 .. setSize                                                  2026   239
2113
2114       ; begin
2115           for i := 0 to setSize                                           2112   239
2116           do irec . ary [ i ] := table [ address + i ]                     2106 2109  2112    *     S  2112
2117       ; result := irec . pset                                             2104 2106  2110
2118         end {ConvertToKeySet}
2119
2120     ; procedure HandleChoice                                               *
2121         ( choicetag : integer ; var choiceTagMatched : Boolean )           *     S    *     S
2122
2123             { This procedure performs input
c 2124            choices.  It sequentially tests each alternative
c 2125            value against the tag value, and when a match is
c 2126            found, performs a branch to the corresponding
c 2127            alternative path.  If none of the alternative
```

```
c 2128              values matches the tag value, table interpretation
c 2129              proceeds to the operation immediately following
c 2130              the list of alternatives (normally the otherwise
c 2131              path).  The flag choicetagMatched is set to true
c 2132              if a match is found and false otherwise.     }
  2133
  2134        ; var numberOfChoices : integer                                  *     S
  2135          ; choiceTableAddress : integer                                 *     S
  2136          ; SSLLineNo : integer                                       2090     S
  2137          ; keyAddress : integer                                         *     S
  2138          ; keys : setofsymbol                                        2090   243
  2139
  2140        ; begin
  2141            choiceTableAddress := table [ tablePointer ]              2135  2116   380
  2142          ; SSLLineNo := table [ tablePointer + 1 ]                   2136  2116   380
  2143          ; keyAddress := table [ tablePointer + 2 ]                  2137  2116   380
  2144          ; numberOfChoices := table [ choiceTableAddress ]           2134  2116  2135
  2145          ; tablePointer := choiceTableAddress + 1                     380  2135
  2146          ; choicetagMatched := false                                  386     S
  2147          ; ConvertToKeySet ( keyAddress , keys )                     2103  2137  2138
  2148          ; Check                                                     2090
  2149              ( SSLLineNo , keys + callStack [ callStackTop ] . sett ) 2136  2138   400   399   247
  2150          ; repeat if table [ tablePointer ] = choicetag              2116   380  2121
  2151              then begin
  2152                choicetagMatched := true                               386     S
  2153              ; tablePointer := table [ tablePointer + 1 ]             380  2116   380
  2154              ; numberOfChoices := 0                                  2134
  2155              end
  2156              else begin
  2157                tablePointer := tablePointer + 2                       380   380
  2158              ; numberOfChoices := numberOfChoices - 1                2134  2134
  2159              end
  2160            until numberOfChoices = 0                                 2134
  2161          ;
  2162          end {HandleChoice}
  2163
  2164          { Execution Tracing }
  2165
  2166      ; procedure Trace                                                   *
  2167
  2168              {imports (operation, table, tablePointer, var IO);}
  2169
  2170        ; begin
  2171                 (IO.PutString ('table index ');
c 2172                  IO.PutInt (tablePointer-1);
c 2173                  IO.PutString ('  Operation ');
c 2174                  IO.PutInt (operation);
c 2175                  IO.PutString ('  Argument ');
c 2176                  IO.PutInt (table[tablePointer]);
c 2177                  IO.PutLine;)
  2178          end {Trace}
  2179
  2180          { Semantic Choice Failure }
  2181
  2182      ; procedure HandleSemanticChoiceFailure                             *
  2183
```

```
  2184              (imports (sourceLineNo, Trace, var IO, Abort);)
  2185
  2186        ; begin
  2187                (IO.PutString ('Semantic choice failed;  Line ');
c 2188                 IO.PutInt (sourceLineNo);
c 2189                 IO.PutLine;
c 2190                 Trace;
c 2191                 Abort;)
  2192          end (HandleSemanticChoiceFailure)
  2193
  2194      ; procedure HandleMatchFailure                                    *
  2195
  2196              { This procedure handles syntax errors in the input
c 2197                to the S/SL program.
c 2198
c 2199                Syntax error recovery:
c 2200                When a mismatch occurs between the the input token
c 2201                and the operand of an input operation,
c 2202          the following recovery is employed.
c 2203
c 2204                The keyAddress is converted to keySet, and keys is
c 2205                created by the union of keySet and the set in the top
c 2206                callStack entry. An error message is emitted.  Incoming
c 2207                tokens are skipped until one is found belonging in keys. }
  2208
  2209        ; var keyAddress : integer                                      2137     S
  2210          ; SSLLineNo : integer  { S/SL sourceLineNo }                  2136     S
  2211          ; keys : setofsymbol                                          2138   243
  2212
  2213        ; begin
  2214            SSLLineNo := table [ tablePointer + 1 ]                     2210  2116   380
  2215          ; keyAddress := table [ tablePointer + 2 ]                    2209  2116   380
  2216          ; ConvertToKeySet ( keyAddress , keys )                       2103  2209  2211
  2217          ; SyntaxError ( sourceLineNo , SSLLineNo )                    2074  2074  2210
  2218          ; skip                                                        1698
  2219              ( keys + callStack [ callStackTop ] . sett                2211   400   399   247
  2220              )                                 { Read input until token in
c 2221                                                  recovery set }
  2222          end (HandleMatchFailure)
  2223
  2224          { Main Walker Program }
  2225
  2226      ; procedure Interpreter                                           *
  2227
  2228                  { Walk the Syntax/Semantic table }
  2229
  2230        ; var keys : setofsymbol                                        2211   243
  2231
  2232        ; begin
  2233            processing := true                                          376     S
  2234          ; tracing := false                                            377     S
  2235          ; semanticizing := true                                       378     S
  2236          ; displaySyntaxErrors := true                                 379     S
  2237          ; tablePointer := 0                                           380
  2238          ; callStackTop := 1                                           399
  2239          ; callStack [ callStackTop ] . sett := [ tEndOfFile ]         400   399   247     *
```

```
2240          ; gError := 0                                                    409
2241          ; while processing                                               376
2242            do begin
2243              operation := table [ tablePointer ]                          381  2116   380
2244            ; tablePointer := tablePointer + 1                             380   380
2245                      { Trace Execution }
2246            ; if tracing                                                   377
2247              then Trace                                                  2166
2248            ; case operation                                               381
2249              of oCall                                                     189
2250                : begin
2251                  if callStackTop < callStackSize                          399   253
2252                  then begin
2253                    ConvertToKeySet                                       2103
2254                      ( table [ tablePointer + 1 ] , keys )               2116   380  2230
2255                  ; callStackTop := callStackTop + 1                       399   399
2256                  ; callStack [ callStackTop ] . sett                      400   399   247
2257                    := keys + callStack [ callStackTop - 1 ] . sett       2230   400   399   247
2258                  ; callStack [ callStackTop ] . returnAddress             400   399   249
2259                    := tablePointer + 2                                    380
2260                  ; tablePointer := table [ tablePointer ]                 380  2116   380
2261                  end
2262                  else begin
2263                    EmitError ( eCallStackOverflow , sourceLineNo )        757   275  2074
2264                  ; processing := false                                    376     S
2265                  ;
2266                  end
2267              end {oCall}
2268            ; oReturn                                                      191
2269              : begin
2270                if callStackTop = 1                                        399
2271                then
2272                              { Return from main S/SL procedure }
2273                  processing := false                                      376     S
2274                else begin
2275                  tablePointer                                             380
2276                  := callStack [ callStackTop ] . returnAddress            400   399   249
2277                ; callStackTop := callStackTop - 1                         399   399
2278                end
2279              end {oReturn}
2280            ; oRepeat , oMerge                                             193   195
2281              : begin
2282                tablePointer := table [ tablePointer ]                     380  2116   380
2283              ;
2284              end {oRepeat , oMerge }
2285            ; oInput                                                       197
2286              : begin
2287                if table [ tablePointer ] = gcursy                        2116   380   422
2288                then
2289                              {AcceptInputToken}
2290                else begin
2291                              { Syntax Error in Input }
2292                  HandleMatchFailure                                      2194
2293                end
2294              ; tablePointer := tablePointer + 3                           380   380
2295              ;
```

```
  2296              end {oInput}                                                                    199
  2297            ; oInputChoice                                                                    *
  2298              : begin
  2299                HandleChoice ( gcursy , choicetagMatched )                                  2120   422   386
  2300              ; if choicetagMatched                                                         386
  2301                then
  2302                                {AcceptInputToken}
  2303              end {oInputChoice}
  2304                        {oChoiceEnd : begin
c 2305                             HandleSemanticChoiceFailure;
c 2306                             end oChoiceEnd;}
  2307                       { The Following are Pass dependent
c 2308                         Data Structure Semantic Operations  }
  2309            ; oSetupGcurPosNc                                                                 *
  2310              : begin
  2311                setup ( gcurposnc , false )                                                 1532   423     S
  2312              end {oSetupGcurPosNc}
  2313            ; oSetupPos                                                                       *
  2314              : begin
  2315                setup                                                                       1532
  2316                  ( PosStack [ PosStackTop - table [ tablePointer ] ]                        427   429  2116   380
  2317                  , false )                                                                    S
  2318              ; tablePointer := tablePointer + 1                                            380   380
  2319              end {oSetupPos}
  2320            ; oCcompress                                                                      *
  2321              : begin
  2322                ccompress                                                                   1569
  2323                  ( Markstack                                                                *
  2324                      [ MarkStackTop - table [ tablePointer ] ]                              430  2116   380
  2325                  )
  2326              ; tablePointer := tablePointer + 1                                            380   380
  2327              end {oCcompress}
  2328            ; oPushConstPos                                                                   *
  2329              : begin
  2330                PushConstPos ( table [ tablePointer ] )                                    1468  2116   380
  2331              ; tablePointer := tablePointer + 1                                            380   380
  2332              end {oPushConstPos}
  2333            ; oPushGcurPosNc                                                                  *
  2334              : begin
  2335                PushPos ( gcurposnc )                                                      1482   423
  2336              end {oPushGcurPosNc}
  2337            ; oPushIndentPos                                                                  *
  2338              : begin
  2339                PushPos                                                                    1482
  2340                  ( Indent                                                                 1766
  2341                      ( PosStack                                                            427
  2342                          [ PosStackTop - table [ tablePointer ] ]                          429  2116   380
  2343                      )
  2344                  )
  2345              ; tablePointer := tablePointer + 1                                            380   380
  2346              end {oPushIndentPos}
  2347            ; oPushCopyPos                                                                    *
  2348              : begin
  2349                PushPos                                                                    1482
  2350                  ( PosStack [ PosStackTop - table [ tablePointer ] ]                      427   429  2116   380
  2351                  )
```

```
2352              ; tablePointer := tablePointer + 1                          380   380
2353              end {oPushCopyPos}
2354            ; oPushCopyMark                                               *
2355              : begin
2356                PushMark                                                 1496
2357                  ( MarkStack                                             428
2358                      [ MarkStackTop - table [ tablePointer ] ]           430  2116   380
2359                  )
2360              ; tablePointer := tablePointer + 1                          380   380
2361              end {oPushCopyMark}
2362            ; oPushCurrentTokenMark                                       *
2363              : begin
2364                mark ( Lmark , gcurp )                                     S    385   419
2365              ; PushMark ( Lmark )                                        1496   385
2366              end {oPushCurrentTokenMark}
2367            ; oPushNextTokenMark                                          *
2368              : begin
2369                mark ( Lmark , gnextp )                                    S    385   419
2370              ; PushMark ( Lmark )                                        1496   385
2371              end {oPushNextTokenMark}
2372            ; oPopPosAndMark                                              *
2373              : begin
2374                PopPos                                                   1510
2375              ; PopMark                                                  1521
2376              end {oPopPosAndMark}
2377            ; oPopPos : begin PopPos end {oPopPos}                        *   1510
2378            ; oMoveCmnts : begin movecmnts end {oMoveCmnts}               *   1641
2379            ; oPopMark : begin PopMark end {oPopMark}                     *   1521
2380            end {case}
2381          end {loop}
2382        end  " Interpreter "
2383
2384    ; begin " MAIN BLOCK "
2385    "% IF PASCAL32"
2386        if param [ 1 ] = '8'                                             170
2387        then breakpnt                                                    452
2388      ; open ( input , 1 , opforin )                                     489     S    360
2389      ; open ( output , 2 , opforout )                                   489     S    360
2390    "% END PASCAL32"
2391    "% IF PASCAL"
2392      ; reset ( input )                                                    S     S
2393      ; rewrite ( output )                                                 S     S
2394    "% END PASCAL"
2395      ; initsets                                                       1918
2396      ; initrwtable                                                    1946
2397    { ; initinp   }
2398      ; pos := 1              " INITIALIZATIONS FOR INSYMBOL "          412
2399      ; getirec ( gnextp )                                              781   419
2400      ; initdumirec ( gnextp )                                         1892   419
2401    "% IF SINGLEPRECISION"
2402      ; gnextp ^ . tokno := 2                                           419   345
2403    "% END SINGLEPRECISION"
2404    "% IF MULTPRECISION"
2405      ; multzero ( gnextp ^ . tokno )                                   583   419   345
2406      ; multplint ( gnextp ^ . tokno , 2 , gnextp ^ . tokno )           589   419   345   419   345
2407    "% END MULTPRECISION"
```

```
2408        ; getlrec ( gcurp )                                                781   419
2409        ; initdumlrec ( gcurp )                                           1892   419
2410      "% IF SINGLEPRECISION"
2411        ; gcurp ^ . tokno := 1                                             419   345
2412      "% END SINGLEPRECISION"
2413      "% IF MULTPRECISION"
2414        ; multzero ( gcurp ^ . tokno )                                     583   419   345
2415        ; multplint ( gcurp ^ . tokno , 1 , gnextp ^ . tokno )             589   419   345   419   345
2416      "% END MULTPRECISION"
2417        ; gfirstp := gcurp                                                 419   419
2418        ; gcurp ^ . dlink := gnextp                                        419   332   419
2419        ; gtrailp := nil                                                   419
2420        ; gcompp := gcurp                                                  419   419
2421        ; gcomptrailp := nil                                               419
2422        ; gfreep := nil                                                    420
2423        ; outpos := leftmargin                                             423   301
2424        ; gcurpos := leftmargin                                            423   301
2425        ; gcurposnc := leftmargin                                          423   301
2426        ; gcursy := tUnknown                                               422     *
2427        ; rbracksw := false                                                414     S
2428        ; ellpsissw := false                                               414     S
2429        ; innewline := true                                                413     S
2430        ; cmntstate := nocomment                                           415   415
2431      "% IF SINGLEPRECISION"
2432        ; gaggsize := 0                                                    426
2433        ; gtokno := 2                                                      425
2434        ; nltokno := 2                                                     425
2435        ; outtokno := 0                                                    425
2436      "% END SINGLEPRECISION"
2437      "% IF MULTPRECISION"
2438        ; multzero ( gaggsize )                                            583   426
2439        ; multzero ( gtokno )                                              583   425
2440        ; multplint ( gtokno , 2 , gtokno )                                589   425   425
2441        ; nltokno := gtokno                                                425   425
2442        ; multzero ( outtokno )                                            583   425
2443        ; multzero ( linespace )                                           583   432
2444        ; multplint                                                        589
2445            ( linespace , rightmargin - leftmargin + 1 , linespace )       432   302   301   432
2446        ; multzero ( multrightmargin )                                     583   432
2447        ; multplint ( multrightmargin , rightmargin , multrightmargin )    589   432   302   432
2448      "% END MULTPRECISION"
2449        ; sourceLineNo := 0                                               2074
2450        ; lineoutct := 0                                                   445
2451        ; linerecct := 0                                                   445
2452        ; tokenct := 0                                                     445
2453        ; skipct := 0                                                      445
2454        ; gpos := leftmargin                                               423   301
2455        ; mark ( gmark , gcurp )                                             S   424   419
2456        ; insymbol ( false )                                              1026     S
2457        ; insymbol ( false )                                              1026     S
2458        ; PosStackTop := 0                                                 429
2459        ; MarkStackTop := 0                                                430
2460        ; Interpreter                                                     2226
2461        ; unload                                                          1391
2462        ; writeln ( output )                                                 S     S
2463        ; writeln ( output )                                                 S     S
```

```
2464        ; lineoutct := lineoutct + 2                                445   445
2465      "% IF PASCAL32"
2466        ; close ( input )                                           553     5
2467        ; close ( output )                                          553     5
2468        ; open ( summary , 0 , opforout )                           489   450   360
2469      "% END PASCAL32"
2470      "% IF PASCAL"
2471        ; rewrite ( summary )                                         5   450
2472      "% END PASCAL"
2473        ; reportstat ( sourceLineNo , 'INPUT LINES ' )              712  2074
2474        ; reportstat ( lineoutct , 'OUTPUT LINES' )                 712   445
2475        ; reportstat ( tokenct , 'TOKENS      ' )                   712   445
2476        ; reportstat ( skipct , 'SKIPPED     ' )                    712   445
2477        ; reportstat ( linerecct , 'HEAP RECORDS' )                 712   445
2478      "% IF PASCAL32"
2479        ; close ( summary )                                         553   450
2480      "% END PASCAL32"
2481        end " MAIN BLOCK "
2482 .
```

cross reference * is def - is asg

```
-A-
 accept                  139*  463
 addr                     56*
 address                2104* 2116
 aggsize                 333*  817-  881- 1263- 1316  1353  1380- 1404  1404  1413  1413  1552- 1607  1607  1613  1613
                        1685- 1729- 1910- 1913
 arg                      77*
 arglist                 108*  165
 argseq                  110*
 argtag                   82*   99
 argtype                  97*  108
 ary                    2109* 2116-
 ascii                    51*

-B-
 backspace                67*
 blankline              1295*
 block                   127*  129*
 bool                    100*
 Boolean                 376   377   378   379  2121
 boolean                  57    84    92   100   122   337   338   339   340   367   386   413   414   519   523   622
                         672   681   963  1003  1026  1032  1532  1782  1821  1838  2108*
 booltype                 82*  100
 borrowin                622*  626-  659   667-
 borrowout               622*  631-  644-  656-  664-  667
 bracecomment            415*  944  1046
 breakpnt                452* 2387
 bufferp                 368*  470   474   502   539   543-  563
 bufp                    365*  468   472-  474   475-  475   499-  500-  537   541-  543   544-  544

-C-
 c                       117*  119*  139*  141*  527*  532   533   543   949*  952   953   955   956
 callerror               115*
 callStack               400* 2149  2219  2239  2256  2257  2258  2276
 CallStackEntry          245*  400
 callStackSize           253*  399   400  2251
 callStackTop            399* 2149  2219  2238- 2239  2251  2255- 2255  2256  2257  2258  2270  2276  2277- 2277
 carddevice               63*
 carryin                 592*  596-  600   604   607   608   610   611-  612-  613-
 ccompress              1569* 1810  1849  1858  2322
 char                     39    43    47    90   117   119   139   141   328   343   356   357   369   511   527   571
                         912   949   949  1031
 Check                  2090* 2148
 choiceTableAddress     2135* 2141- 2144  2145
 choicetag              2121* 2150
 choiceTagMatched       2121*
 choicetagMatched        386* 2146- 2152- 2299  2300
 chr                     706   955  1943  1943
 close                   124*  553* 2068  2466  2467  2479
 closed                  360*  564
 closedlist             1833*
 cmntstate               415*  944-  945- 1046  2430-
 codelimit               114*
```

complete 70*
concode 51*
control 65*
ConvertToKeySet 2103* 2147 2216 2253
cr 34*
currentchar 369* 463 464 474= 477= 479 480= 481 482= 486= 514 516

-D-
digit 695* 705= 706
digits 442* 992* 995 1022 1066 1103 1930= 1932 1937
diskdevice 62*
Display 776
display 141* 532 735 738 741 752 1715 1727 1762 2086
DisplayInteger 725* 775 1714 1725 1726 1758 1760 1761 2080 2082 2084 2085
DisplayString 744* 761 764 766 768 770 772 774 1711 1724 1757 1759 2079 2081 2083
displaySyntaxErrors 379* 1722 1755 2077 2236=
displaysyntaxErrors 1709
dlink 332* 786 792= 1232= 1243= 1258 1267 1273 1288 1313= 1319= 1350= 1360= 1387 1454 1540
1541 1602 1622 1630 1653 1654 1656 1659= 1674 1680= 1681= 1730 1897= 2418=
dummyLine 237*

-E-
eCallStackOverflow 275* 771 2263
eEndOfErrors 263*
eExtraneousProgramText 277*
element 1468* 1474 1482* 1488
elementhandler 1781* 1820* 1829 1837* 1852
ellpsissw 414* 1141 1147= 2428=
em 35* 479 558 562
eMarkStackOverflow 283* 767 1505
eMarkStackUnderflow 285* 769 1527
EmitError 757* 1477 1491 1505 1516 1527 2263
empty 51*
endfile 70*
endmedium 71*
eNoError 261*
eof 519* 521= 899 1050 1073 1077 2035 2039 2045 2047 2056
eoff 367* 459 480= 504= 521
eoln 523* 525= 901 940 943 1043 1052 1074 1078 1217 2040 2048 2057
eolnf 367* 482= 483= 505= 525
ePosStackOverflow 279* 763 1477 1491
ePosStackUnderflow 281* 765 1516
ePrematureEndOfFile 273*
errorCode 758* 762 778
ErrorCodes 289* 758
eSyntaxError 269*
exitloop 963* 974= 976= 982= 988 1032* 1048= 1053= 1212= 1224

-F-
f 122* 124* 127* 129* 131* 455* 458 490* 493 508 511* 514 516 519* 521 523*
525 527* 530 549* 551 553* 556 562 571* 574 576
failure 70*
false 483 504 505 626 644 777 833 844 974 1053 1147 1212 1245 1478 1492 1506
1517 1528 1564 1565 1628 1848 1899 2109* 2146 2234 2264 2273 2311 2317 2427 2428
2456 2457
fc 912* 925

ff 33* 311* 315* 1943
fileattr 53*
filekind 51* 55
filestatus 360* 366 490
filler1 91*
filler2 92*
filler3 94*
fint 693* 698
firstChoiceOperation 225*
firstdotpos 412* 1143
firstEmittingOperation 229*
firstErrorCode 259* 289
firstFatalErrorCode 271*
firstInputToken 295*
firstParameterizedOper- 221*
ation
firstParserErrorCode 267*
firstPrimitiveOperatio- 187*
n
firstSemanticOperation 215*
firstSimpleOperation 217*
firsttableOperation 185* 381
firstTrueErrorCode 265*
flmovecmnts 1782* 1799 1807 1821* 1830 1838* 1853
flp 1003* 1007
fnewline 1532* 1551
fno 364* 460 462 470 506- 531 539 557 559 563
fnum 712* 718
forward 803*
found 122*
fpos 1532* 1543 1550 1766* 1771 1778* 1797 1798 1805 1817* 1829 1834* 1848 1852 1856
fraglength 307* 343 917 1445
freelrec 789* 1458
fstr 712* 721
fstring 693* 700- 702- 706- 1861* 1869 1873 1875 1883
fsymbol 1861* 1887
fsys 1698* 1713 1717 1730

-G-
gaggsize 426* 817 827- 827 830 830 839- 839 842 842 850- 851 855 855 864- 864
868 869 881 883- 883 886 886 1035 1263 1322- 1322 1326 1326 1362- 1362 1366
1366 1552 1554- 1554 1557 1557 1651 1685 1687 1691 1708 1729 1733- 1733 1736 1736
1744 1745 1748 1750 2432- 2438
gcompp 419* 1251- 1259 1261- 1328 1330- 1336 1345 1350 1353 1354 1368- 1404 1405 1407 1412
1413 1416 1422 1462 1581 1586 1607 1610 1613 1615 1619 1632 1660 1661- 1675 1677-
1682 1683- 2420-
gcomptrallp 419* 1261- 1331- 1357- 1360 1464- 1677- 2421-
gcurp 419* 814 879 1251 1252 1254 1257 1258- 1258 1261 1263 1265 1266 1308 1313 1316
1317 1328 1462 1537 1632 1652 1653 1654 1656- 1656 1658 1659 1661 1663 1663 1665
1673 1674- 1674 1677 1680 1682 1684- 1685 1719 1761 1795 1803 2085 2364 2408 2409
2411 2414 2415 2417 2418 2420 2455
gcurpos 423* 822 824 825 832- 837 845- 858- 871- 889 891- 892- 1559- 1563 1633- 1740-
1741- 1742 2424-
gcurposnc 423* 1563- 1634- 1742- 1805 2311 2335 2425-
gcursy 422* 1036 1265- 1717 1725 1760 1799 1801 1804 1807 1846 1855 2084 2096 2287 2299
2426-

gError 409* 778- 2240-
get 455* 508 908 2034 2043 2054 2059
getIrec 781* 919 1039 1301 1338 2399 2408
gfirstp 419* 1317 1318- 1354 1356- 1404 1405 1412 1413 1421 1461- 1604 1657 1658- 2417-
gfreep 420* 784 786 786- 792 793- 2422-
gmark 424* 2455
gnextp 419* 1038 1252 1264- 1268 1653 1663 2369 2399 2400 2402 2405 2406 2406 2415 2418
gpos 423* 2454-
gtokno 425* 1234- 1234 1237 1237 1239 1303- 1303 1306 1306 1340- 1340 1343 1343 2433- 2439
2440 2440 2441
gtrailp 419* 1257- 1261 1319 1320- 1358- 1463- 1659 1673- 1677 1681 2419-

-H-
HandleChoice 2120* 2299
HandleMatchFailure 2194* 2292
HandleSemanticChoiceFa- 2182*
ilure
header 137*
heaplimit 114*
hexdigits 442* 1931-
hi 962* 973- 975 978 986-
ht 310* 314* 1943

-I-
i 418* 585* 587- 587 592* 597- 599 600 602 605 607 607 608 608 609 610
621* 627- 629 629 632 632 634 636 638 640 640 640 642 642 645 645
647 649 651 653 653 653 656 656 657 657 657 660 661 663 665 665
699- 703 706 707- 707 715* 719- 719 721- 721 727* 732- 733 733 734- 734
736 738 739- 739 746* 749- 750 750 752 753- 753 917 923- 925 926- 926
1023- 1041- 1395* 1442- 1445 1446- 1447 1448- 1448 1701* 1712- 1713 1714 1768* 1771- 1772
1773 1863* 1871- 1873 1873 1874- 1874 1875 1878 1883 1887 1950- 1950 2026* 2038- 2050
2052 2053- 2053 2061 2061 2112* 2115- 2116 2116
id 102* 122* 164*
identifier 47* 102 122 164
identify 137*
idlength 45* 47
idtype 82* 102*
incharacter 896* 934 1044 1053 1211
include 178*
indent 1766* 1773- 1774- 1797 2340
indentamt 306* 1771
indentf 1782* 1796 1821* 1830
initdumirec 1892* 2400 2409
initfile 173* 447* 450* 2030 2032 2034 2035 2039 2040 2041 2043 2045 2047 2048 2049 2052
2054 2056 2057 2058 2059 2068
initinp 2023*
initrw 1861* 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965
1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981
1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997
1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013
2014 2015 2016 2017 2018 2019 2020
initrwtable 1946* 2396
initsets 1918* 2395
innewline 413* 905- 1231 1245- 2429-
inp 110*
input 65* 173* 450* 899 901 908 933 940 941 943 995 996 1019 1022 1043 1050

	1052	1054	1055	1065	1073	1074	1075	1077	1078	1079	1082	1094	1107	1117	1132	1151
	1173	1181	1193	1217	1218	1241	2388	2392	2466							
inputtask	80*															
inputwindow	511*	515=	516=	571*	575=	576=	933	941	995	996	1019	1022	1054	1055	1065	1075
	1079	1082	1094	1107	1117	1132	1151	1173	1181	1193	1218	1241	2041	2049	2052	2058
inropenlist	1777*	1828	1851													
insertchar	912*	933	1144	1225												
insymbol	1026*	1565	1753	2456	2457											
int	101*															
int2string	693*	718	731													
integer	20	26*	56	58	91	93	101	127	129	131	133	135	166	320	322	325
	365	407	408	411	418	441	445	590	592	621	693	695	712	715	725	727
	746	758	962	1395	1534	1701	1768	1863	2026	2074	2090	2104	2121	2134	2135	2136
	2137	2209	2210													
integer32	20*															
interpreter	2226*	2460														
intervention	70*															
inttype	82*	101*														
ioarg	67*	77														
iodevice	61*															
iooperation	65*	75														
iop	590*	596														
ioparam	73*															
ioresult	69*	76														
-J-																
j	962*	970=	971	971	1863*	1872=	1873	1877=	1878	1880	1880	1881=	1881	1884=	1885	1885
	2026*	2061=	2062													
jobtask	80*															
-K-																
keyAddress	2137*	2143=	2147	2209*	2215=	2216										
keys	384*	2090*	2096	2099	2138*	2147	2149	2211*	2216	2219	2230*	2254	2257			
kind	55*															
-L-																
laggsize	1029*	1035=	1298*	1316=	1353=	1380	1382=	1382	1385	1385	1644*	1651=	1687	1691	1700*	1708=
	1744	1745=	1748	1750=												
lastChoiceOperation	227*															
lastEmittingOperation	231*	233														
lastErrorCode	287*	289														
lastInputToken	299*															
lastParameterizedOpera-tion	223*															
lastPrimitiveOperation	207*															
lastSemanticOperation	233*	235														
lastSimpleOperation	219*															
lasttableOperation	235*	381														
lc	1031*															
lcurp	1643*	1652=	1657	1660	1662	1679	1680	1681	1683	1684						
leftmargin	301*	864	868	1314	1322	1326	1351	1362	1366	1414	1546	1547	1687	1744	1907	2423
	2424	2425	2445	2454												
len	1863*	1868=	1869	1869	1870=	1870	1871	1872	1884							
length	131*	336*	816=	816	820	821	824	827	830	832	839	842	845	852	855	858
	883	886	890	892	1228=	1276	1314=	1351=	1382	1385	1443	1450	1539=	1539	1541	1541
	1542	1554	1557	1559	1601	1625	1731=	1731	1733	1736	1739	1741	1908=			

letters 442* 1066 1921= 1936 2041 2049 2058
lexchars 442* 1054 1219 1935=
line 43* 137 166* 170
linelength 41* 43
lineoutct 445* 1431= 1431 2450= 2464= 2464 2474
linep 345* 809= 1601 1601 1602
linerec 329 330*
linerecct 445* 785= 785 2451= 2477
linerecptr 329* 332 345 417 421 781 789 805 914 1003 1030 1297 1394 1572 1643 1892
linespace 432* 1406 1692 1749 2443 2445 2445
lint 695* 698= 701 703 705 708= 708
llength 1534* 1541= 1542= 1543 1545
llp 1572* 1602= 1604= 1607 1610 1613 1615 1619 1621 1622 1624 1625 1626 1628 1630= 1630
llp1 1030* 1266= 1267 1276 1276 1287= 1394* 1421= 1422 1424 1425 1426 1430 1433 1436 1441
1443 1450 1452 1453 1454= 1454 1461
llp2 1030* 1267= 1268 1270 1281 1287 1288= 1288 1394* 1441= 1446= 1446 1447 1453= 1455 1456
1457= 1457 1459
llp3 1394* 1456= 1458
llp4 1394* 1455= 1459
Lmark 385* 2364 2365 2369 2370
lmult 1397* 1404 1406 1574* 1580 1581 1607 1608 1608 1609 1646* 1691 1692 1703* 1748 1749
lnextp 1030* 1038= 1243 1244= 1264
lnumstr 714* 718 719 728* 731 733 738
lo 962* 972= 975 978 985=
logicalunitno 359* 364 490
lookupident 959* 1068
lpos 1393* 1443= 1571* 1600= 1604= 1608 1613 1622= 1624 1625= 1625 1627= 1627 1633 1634
lpos1 1788* 1797= 1798= 1806
lpos2 1788* 1805= 1806=
lrec 2106* 2116 2117
lrp 1892* 1895 1898
lstart 1534* 1545= 1546 1548
lstring 961* 969= 971= 979 984 2025* 2037= 2052= 2063
lsy 1028* 1194= 1195= 1196= 1197= 1201= 1202= 1203= 1204= 1207
lsys 1789* 1793= 1794
ltokno 1298* 1308= 1345= 1373 1375= 1375 1378 1378 1644* 1662= 1665 1667= 1667 1670 1670 1679
lu 490* 495 506

-M-

m 805* 808 1496* 1502 1569* 1580 1582 1586 1587 1592 1596 1601 1601 1602
mark 133* 805* 1795 1803 1825* 1830 1842* 1854 1858 2364 2369 2455
mark1 1787* 1795 1810 1811= 1813
mark2 1787* 1803 1811
markrec 345* 385 424 428 805 1496 1569 1779 1784 1787 1818 1825 1835 1842
MarkStack 428* 1502= 2357
Markstack 2323
MarkStackTop 430* 1499 1501= 1501 1502 1524 1525= 1525 2324 2358 2459=
maxarg 106* 108
maxErrors 291* 409
maxint 22 28* 600 602 605 632 634 637 645 647 650 660 661
maxint32 22*
maxLineNumber 255* 383
MaxMarkStack 304* 428 430 1499
MaxPosStack 303* 427 429 1471 1485
maxshortint 28
mode 490* 498 500 501

mop 590* 599 600 602 607 608 609 610
move 65*
moveable 340* 1281- 1284- 1654 1663
movecmnts 1641* 1800 1808 2378
multge 672* 678- 1582 1609 1692
multgt 681* 688- 1406 1581 1596 1610 1749
multint 322* 325* 333 345 425 426 432 583 590 590 593 619 619 623 672 674
681 683 1029 1298 1397 1574 1644 1646 1700 1703
multmct 317* 325 585 587 597 627 678 688
multplint 589* 686 830 842 855 867 886 1237 1306 1325 1343 1365 1378 1385 1557 1580
1608 1670 1736 2406 2415 2440 2444 2447
multrightmargin 432* 1609 2446 2447 2447
multsubtr 618* 677 687 1403 1606 1691 1748
multzero 583* 798 1904 1913 2405 2414 2438 2439 2442 2443 2446

-N-
new 502 785
newlength 411* 927- 927 966 970 972 973 1042- 1228
newline 337* 833- 844- 857- 862- 1231- 1272 1273 1424 1551- 1628- 1899-
newp 417* 920 921 922- 925 971 1039 1040 1040 1226 1243 1244
newstartpos 412* 1045- 1049- 1143- 1230
newsymbol 416* 946- 967- 976- 981- 998- 1051- 1084- 1089- 1097- 1099- 1110- 1112- 1120- 1122- 1127-
1135- 1137- 1146- 1154- 1156- 1163- 1168- 1176- 1184- 1186- 1207- 1222- 1229 1246
niltype 82* 100
nl 32* 481 551 776 1715 1727 1762 2086
niltokno 425* 1430- 1582 1587 2434- 2441-
nocomment 415* 945 2430
notused 58* 143*
notused10 161*
notused2 145*
notused3 147*
notused4 149*
notused5 151*
notused6 153*
notused7 155*
notused8 157*
notused9 159*
Number 725* 731
number 992* 1103
numberOfChoices 2134* 2144- 2154- 2158- 2158 2160

-O-
oCall 189* 2249
oCcompress 2320
oChoiceEnd 205*
octaldigits 442* 1928- 1930
oEmit 201*
oError 203*
oInput 197* 2285
oInputChoice 199* 2297
oMerge 195* 2280
oMoveCmnts 2378
op1 619* 629 632 634 640 642 645 647 653 656 657 672* 677 681* 687
op2 619* 629 632 636 640 642 645 649 653 656 657 672* 677 681* 686
open 121* 489* 2030 2388 2389 2468
openlist 1816*

operation 75* 381* 2243- 2248
opforin 360* 466 498 2030 2388
opforout 360* 500 535 560 2389 2468
oPopMark 2379
oPopPos 2377
oPopPosAndMark 2372
options 90*
oPushConstPos 2328
oPushCopyMark 2354
oPushCopyPos 2347
oPushCurrentTokenMark 2362
oPushGcurPosNc 2333
oPushIndentPos 2337
oPushNextTokenMark 2367
ord 706 952 952 953 953 955 955 955
oRepeat 193* 2280
oReturn 191* 2268
oSetupGcurPosNc 2309
oSetupPos 2313
out 110*
outpos 423* 1425 1429- 1434- 1436 1439- 1439 1450- 1450 1604 2423-
output 65* 173* 450* 1428 1438 1447 2389 2393 2462 2463 2467
outputtask 80*
outtokno 425* 1452- 1592 1596 2435- 2442
overflow 113*

-P-
p 127* 129* 170* 781* 785 786- 786 789* 792 793 795 798 805* 808 808 914*
919 920 921 922 1297* 1301 1309 1312 1318 1319 1320 1330 1338 1346 1349 1356
1357 1358 1360 1368 1370 1371 1387-
packed 343 356
page 39* 127 129 361
pagelength 37* 39 468 499 537
pageno 365* 470 471- 471 497- 539 540- 540 563
pageptr 361* 368
parlcase 350*
paractparam 349*
param 165* 170* 2386
parameterValue 407*
parblock 352*
parcompoundstmt 350*
parconstant 347*
parexpr 349*
parfactelem 348*
parfactor 348*
parfidlist 351*
parfparm 350*
parpfhead 351*
parproc 346* 1781 1820 1837
parproghead 351*
parprogram 353*
parsetelem 348*
parsmpexpr 349*
parsmptype 351*
parstc 347*
parstcelem 347*

parstcval 347*
parstmt 350*
parsyscomp 352*
partcompress 1782* 1809 1821* 1830 1838* 1853 1857
parterm 349*
partype 352*
partypenamer 352*
parvariable 348*
parxstmt 350*
passlink 86 88*
passptr 86* 103
pfxfile 49* 122 124 127 129 131
pfxget 126* 470
pfxread 117* 464
pfxwrite 119* 533 558
pointer 84* 94
pointererror 113*
PopMark 1521* 2375 2379
PopPos 1510* 2374 2377
pos 412* 903- 907- 907 1045 1049 2398-
PositIontyp 1482
positIontyp 320* 335 336 412 423 427 1393 1468 1532 1571 1766 1766 1778 1788 1817 1834
PosStack 427* 1474- 1488- 2316 2341 2350
PosStackTop 429* 1471 1473- 1473 1474 1485 1487- 1487 1488 1513 1514- 1514 2316 2342 2350 2458-
ppp 173
prbracecmnt 937* 1047 1159
pred 665 707 1228 1870 1881
prepcommand 1003* 1009- 1010- 1281
printdevice 62*
probe 962* 978- 979 981 984 985 986
processDollarSign 1015* 1080 1174
processing 376* 777- 1478- 1492- 1506- 1517- 1528- 2233- 2241 2264- 2273-
progresult 112* 167
protected 57*
pset 2110* 2117
ptr 103*
ptrtype 82* 103*
PushConstPos 1468* 2330
PushMark 1496* 2356 2365 2370
PushPos 1482* 2335 2339 2349
put 129* 539 563

-R-

rangeerror 113*
rbracksw 414* 2427-
release 135*
reportstat 712* 2473 2474 2475 2476 2477
reset 2032 2392
resetPoint 93*
result 167* 583* 587- 590* 615- 619* 669- 2104* 2117-
resultValue 408*
returnAddress 249* 2258- 2276
returnmark 1784* 1813-
rewind 67*
rewrite 2393 2471
rightmargin 302* 820 821 824 864 868 871 890 891 1314 1322 1326 1351 1362 1366 1414

1434 1543 1545 1614 1687 1739 1740 1744 1772 1774 2445 2447
rlink 332* 920- 920 921- 1007 1040- 1312- 1349- 1441 1446 1457 1898-
run 163*
rw 435* 979 981 984 1873 1875 1880- 1880 1883 1887 2062
rwct 441* 1866 1888- 1888 1949-
rwmax 308* 434 436 1866
rwptr 434* 972 973 1871 1872 1877 1885- 1885 1950- 2061 2061

-S-
s 744* 750 752
scratch 51*
semanticizing 378* 2235-
sepafter 339* 1240- 1730
sepsys 1780* 1792 1799 1804 1807 1819* 1829 1836* 1845 1852
seqcode 51*
SetArray 241*
setArray 2109
setofchar 328* 444 992
setofsymbol 243* 247 384 1698 1780 1783 1789 1819 1822 1836 1839 2090 2104 2110 2138 2211
2230
setSize 239* 241 2112 2115
sett 247* 2149 2219 2239- 2256- 2257
setup 1532* 1805 1848 1856 2311 2315
setupchar 930* 942 945 995 997 997 1018 1021 1023 1067 1072 1081 1083 1088 1093 1096
1106 1109 1116 1119 1126 1131 1134 1145 1150 1153 1159 1162 1167 1172 1175 1180
1183 1206 1221
setupcmnt 811* 1256 1672
shiftpos 305*
shortinteger 26 241
skip 338* 1564- 1626 1698* 1721- 1794 1845 2099 2218
skipcmnt 876* 1256
skipct 445* 1738- 1738 2453- 2476
skipping 1026* 1256
sourceLineNo 383* 758* 775 904- 904 1477 1491 1505 1516 1527 1758 2074* 2080 2098 2217 2263
2449- 2473
SSLLineNo 2074* 2082 2090* 2098 2136* 2142- 2149 2210* 2214- 2217
stacklimit 114*
startmark 1779* 1818* 1829 1835* 1849 1852
startmedium 71*
startpos 335* 820 821- 822 825- 832 837- 845 858 889- 890 892 1230- 1276 1277 1425
1436 1547- 1548- 1550- 1559 1601 1622 1624- 1739 1741 1907-
status 76* 366* 466 501- 535 560 564-
stopsys 1783* 1793 1822* 1830 1839* 1845 1853
string 438* 979 984 1873 1875 1883- 2063
string12 354* 438 693 712 714 728 961 1861 2025
StringType 744
stringType 357*
succ 471 475 540 544 734 739 785 816 904 907 926 927 1234 1250 1303 1340
1375 1431 1439 1448 1539 1627 1667 1731 1738 1874 1885 1888 2053
summary 173* 447* 450* 719 720 721 722 2468 2471 2479
symbol 334* 439* 818 835 847 860 888 981 1229- 1254 1259 1265 1274- 1275- 1278- 1279-
1311- 1336 1348- 1407 1416 1426 1433 1540 1621 1675 1730 1887- 1906- 2064-
symboltyp 242* 243 334 416 422 439 1028 1861
SyntaxError 2074* 2098 2217
sysclose 1836* 1845 1853 1855
syselemdeq 1780* 1792 1819* 1829 1836* 1845 1852

sysloop 1789* 1792= 1793 1801
sysopen 1836* 1845 1846

-T-
table 2116 2141 2142 2143 2144 2150 2153 2214 2215 2243 2254 2260 2282 2287 2316 2324
2330 2342 2350 2358
tablePointer 380* 2141 2142 2143 2145= 2150 2153= 2153 2157= 2157 2214 2215 2237= 2243 2244= 2244
2254 2259 2260= 2260 2275= 2282= 2282 2287 2294= 2294 2316 2318= 2318 2324 2326= 2326
2330 2331= 2331 2342 2345= 2345 2350 2352= 2352 2358 2360= 2360
tableSize 249 380
tabstraction 1951
tag 99*
taligned 1952
tall 1953
tand 1954
tany 1955
tapedevice 62*
tarray 1956
taskkind 80*
tassert 1957
tassign 1135
tasterisk 1168
tat 1958
tbegin 1959
tbind 1960
tbits 1961
tblankline 1311 1336 1348 1426 1433
tbound 1962
tcase 1963
tchar 1176
tcheckable 1964
tchecked 1965
tcode 1966
tcollection 1967
tcolon 1137
tcomma 1197
tcomment1 860 946 1247 1255 1259 1274 1675
tcomment2 818 1247 1255 1279 1621
tcomment3 835 888 1247 1255 1278 1540
tcomment4 847 1248 1255 1259 1275 1621 1675
tconst 1968
tconverter 1969
tcounted 1970
tdecreasing 1971
tdefault 1972
tdependent 1973
tdiv 1974
tdoubleperiod 1146 1154
tdummy 1906
telse 1975
telself 1976
tempresult 593* 605= 607= 608= 615 623* 638= 640= 651= 653= 657= 660 661= 663 665= 665
669
tend 1977
tEndOfFile 2239
tendoffile 1036 1051 1407 1416

```
tequal                  1186
terminated               113*
texit                   1978
texports                1979
text                     341*  362*  447   450   455   490   511   519   523   527   549   553   571   925=  971   1008
                        1009  1010  1447
texternal               1980
tfinally                1981
tfor                    1982
tforward                1983
tfrom                   1984
tfunction               1985
tgreaterequal           1120
tgreaterthan            1122
throwaways               443*  514   574  1241  1943=
tident                   967   976  2064
tif                     1986
timellimit               114*
timplies                1097
timports                1987
tin                     1988
tinclude                1989
tinitially              1990
tinline                 1991
tinvariant              1992
tlabel                  1184
tleftparen              1127
tlessequal              1110
tlessthan               1112
tloop                   1993
tmachine                1994
tminus                  1099
tmod                    1995
tmodule                 1996
tnot                    1997
tnumber                  998
tof                     1998
tokenct                  445*  1250= 1250  2452= 2475
tokno                    333*   345*  795=  798   808=  808  1239= 1308  1345  1373= 1430  1452  1580  1581  1582  1586
                        1586   1587  1592  1596  1610  1610  1615  1615  1662  1665= 1679= 1726  1761  1901= 1904  2085
                        2402= 2405  2406  2406  2411= 2414  2415  2415
top                      133*   135*
tor                     1999
totherwise              2000
tpacked                 2001
tparameter              2002
tperiod                 1156
tpervasive              2003
tplus                   1089
tpost                   2004
tpre                    2005
tprocedure              2006
Trace                   2166* 2247
tracing                  377* 2234= 2246
transmission              70*
treadonly               2007
```

```
 trecord                2008
 treturn                2009
 treturns               2010
 trightparen            1163
 true                    480   482   631   664   857   862   905   976   982  1048  1284  1721  1753  1805  1853  1856
                        2110* 2152  2233  2235  2236  2429
 tsemicolon             1202
 tset                   2011
 tstring                1084
 tSyntaxError            297*
 tthen                  2012
 tthus                  2013
 tto                    2014
 ttype                  2015
 tUnknown               2426
 tunknown               1194  1195  1196  1203  1204  1222  2016
 tuparrow               1201
 tvar                   2017
 twhen                  2018
 twith                  2019
 tXor                    299
 txor                   2020
 typedevice               62*

-U-
 unload                  802* 1391* 1637  1688  1693  1745  1750  2461
 upperCase               949*  954=  956=  971
 upspace                  67*

-V-
 varianterror            114*

-W-
 work                    592*  602=  603=  603   604=  604   605   621*  634=  635=  635   636=  636   637=  637   638
                         647=  648=  648   649=  649   650=  650   651   674*  677   678   683*  686   687   687   688
 write                   527*  551   562   719   720   721  1438  1447
 writeeof                 67*
 writeln                 549*  722  1428  2462  2463

end xref  595 identifiers  3185 total references
2328 collisions.
```

Appendix 4

Syntax/Semantic Language code

```
 1
 2
 3
 4 %       Project Euclid                                  *     *
 5 %
 6 %       Euclid Parser Tables V1.24                      *     *     *     *
 7 %       Author: J.R. Cordy                              *     *     *     *
 8 %       Reader: D.R. Crowe                              *     *     7     *
 9 %       Date:   27 Feb 1979                             *     *
10
11
12
13 Inputs :                                                *
14
15     tIdent       'Ident'                                *
16     tNumber      'Number'                               *
17     tOctalNumber 'OctalNumber'                          *
18     tHexNumber   'HexNumber'                            *
19     tString      'String'                               *
20     tExString    'ExString'                             *
21     tMDString    'MDString'                             *
22     tChar        'Char'                                 *
23     tExChar      'ExChar'                               *
24     tMDChar      'MDChar'                               *
25     tComment1                                           *
26     tComment2                                           *
27     tComment3                                           *
28     tComment4                                           *
29     tCodeStart   'CodeStart'                            *
30     tCodeMiddle  'CodeMiddle'                           *
31     tCodeEnd     'CodeEnd'                              *
32     tIllegal                                            *
33     tNewLine     '<NL>'                                 *
34     tEndOfFile                                          *
35     tPlus        '+'                                    *
36     tMinus       '-'                                    *
37     tAsterisk    '*'                                    *
38     tEqual       '='                                    *
39     tLessThan    '<'                                    *
40     tGreaterThan '>'                                    *
41     tLessEqual   '<='                                   *
42     tGreaterEqual '>='                                  *
43     tImplies     '->'                                   *
44     tLeftParen   '('                                    *
45     tRightParen  ')'                                    *
46     tPeriod      '.'                                    *
47     tDoublePeriod '..'                                  *
48     tAssign      ':='                                   *
49     tComma       ','                                    *
50     tSemicolon   ';'                                    *
51     tColon       ':'                                    *
52     tUpArrow     '^'                                    *
53     tLabel       '=>'                                   *
54     tAbstraction 'abstraction'                          *
55     tAligned     'aligned'                              *
```

```
 56      tAll          'all'                                              *
 57      tAnd          'and'                                              *
 58      tAny          'any'                                              *
 59      tArray        'array'                                            *
 60      tAssert       'assert'                                           *
 61      tAt           'at'                                               *
 62      tBegin        'begin'                                            *
 63      tBind         'bind'                                             *
 64      tBits         'bits'                                             *
 65      tBlankline    'blankline'                                        *
 66      tBound        'bound'                                            *
 67      tCase         'case'                                             *
 68      tCheckable    'checkable'                                        *
 69      tChecked      'checked'                                          *
 70      tCode         'code'                                             *
 71      tCollection   'collection'                                       *
 72      tConst        'const'                                            *
 73      tConverter    'converter'                                        *
 74      tCounted      'counted'                                          *
 75      tDecreasing   'decreasing'                                       *
 76      tDefault      'default'                                          *
 77      tDependent    'dependent'                                        *
 78      tDiv          'div'                                              *
 79      tDummy        'dummy'                                            *
 80      tElse         'else'                                             *
 81      tElseif       'elseif'                                           *
 82      tEnd          'end'                                              *
 83      tExit         'exit'                                             *
 84      tExports      'exports'                                          *
 85      tExternal     'external'                                         *
 86      tFinally      'finally'                                          *
 87      tFor          'for'                                              *
 88      tForward      'forward'                                          *
 89      tFrom         'from'                                             *
 90      tFunction     'function'                                         *
 91      tIf           'if'                                               *
 92      tImports      'imports'                                          *
 93      tIn           'in'                                               *
 94      tInclude      'include'                                          *
 95      tInitially    'initially'                                        *
 96      tInline       'inline'                                           *
 97      tInvariant    'invariant'                                        *
 98      tLoop         'loop'                                             *
 99      tMachine      'machine'                                          *
100      tMod          'mod'                                              *
101      tModule       'module'                                           *
102      tNot          'not'                                              *
103      tOf           'of'                                               *
104      tOr           'or'                                               *
105      tOtherwise    'otherwise'                                        *
106      tPacked       'packed'                                           *
107      tParameter    'parameter'                                        *
108      tPervasive    'pervasive'                                        *
109      tPost         'post'                                             *
110      tPre          'pre'                                              *
111      tProcedure    'procedure'                                        *
```

```
112     tReadonly    'readonly'
113     tRecord      'record'
114     tReturn      'return'
115     tReturns     'returns'
116     tSet         'set'
117     tThen        'then'
118     tThus        'thus'
119     tTo          'to'
120     tType        'type'
121     tUnknown     'unknown'
122     tVar         'var'
123     tWhen        'when'
124     tWith        'with'
125     tXor         'xor';
126
```

```
127 SimpleOps :                                                    *
128
129     oSetupGcurPosNc                                            *
130     oPushGcurPosNc                                             *
131     oPushCurrentTokenMark                                      *
132     oPushNextTokenMark                                         *
133     oPopPosAndMark                                             *
134     oPopPos                                                    *
135     oPopMark                                                   *
136     oMoveCmnts;                                                *
137 ParmOps :                                                      *
138
139     oPushIndentPos                                             *
140     oPushConstPos                                              *
141     oPushCopyPos                                               *
142     oPushCopyMark                                              *
143     oSetupPos                                                  *
144     oCcompress;                                                *
145 _|_
146
```

```
147
148 %    Programs                                                    *
149
150 Programs :                                                       148
151
152     oPushConstPos(1)                                            140
153     oPushCurrentTokenMark                                       131
154     oPushCopyPos(0)                                             141
155     oPushCopyMark(0)                                            142
156     ∂Pervasive                                                   *
157     'type'
158     oPushCurrentTokenMark                                       153
159     oSetupGcurPosNc                                             129
160     oPushIndentPos(0)                                           139
161     ∂TypeDeclaration                                             *
162     oCcompress(0)                                               144
163     oPopMark                                                    135
164     ';'
165     oMoveCmnts                                                  136
166     oSetupPos(0)                                                143
167     oPopPos;                                                    134
168
169
170
171 %    Modules                                                     *
172
173 ModuleBody :                                                     *
174     [=
175         | 'Ident':
176             oSetupPos(0)                                        166
177             oCcompress(0)                                       162
178         | *:
179     ]
180     oPushCurrentTokenMark                                       158
181     [[=
c 182         |'Imports':
c 183             oPushCurrentTokenMark
c 184             oPushIndentPos(0)
c 185             oSetupPos(0)
c 186             oPushNextTokenMark
c 187             oPushIndentPos(0)
c 188             ∂ImportClause
c 189             oCcompress(0)
c 190             oPopPosAndMark
c 191             ';'
c 192             oMoveCmnts
c 193             oSetupPos(0)
c 194         |*:
c 195             >
c 196     ]]
197     [=
198         |'exports':
199             oPushCurrentTokenMark                               158
200             oPushIndentPos(0)                                   160
```

```
201             oSetupPos(0)                                    166
202             oPushNextTokenMark                              132
203             oPushIndentPos(0)                               160
204             @ExportClause                                     *
205             oCcompress(0)                                   177
206             oPopPosAndMark                                  133
207             ';'
208             oMoveCmnts                                      165
209             oSetupPos(0)                                    201
210         |*:
211     ]
212     [=
213         |'not':
214             oPushIndentPos(0)                               203
215             oSetupPos(0)                                    209
216             'checked'
217             oSetupGcurPosNc                                 159
218             ';'
219             oPopPos                                         167
220             oMoveCmnts                                      208
221             oSetupPos(0)                                    215
222         |'checked':
223             oPushIndentPos(0)                               203
224             oSetupPos(0)                                    221
225             ';'
226             oPopPos                                         219
227             oMoveCmnts                                      220
228             oSetupPos(0)                                    224
229         |*:
230     ]
231     oPushCopyPos(0)                                         154
232     oPushNextTokenMark                                      202
233     @Declarations                                             *
234     [=
235         |'initially':
236             oPushIndentPos(0)                               203
237             oPushCurrentTokenMark                           158
238             oSetupPos(0)                                    228
239             oPushNextTokenMark                              232
240             oPushCopyPos(0)                                 154
241             @RoutineDefinition                                *
242             oCcompress(0)                                   205
243             oPopPosAndMark                                  206
244             ';'
245             oMoveCmnts                                      227
246             oSetupPos(0)                                    238
247         |*:
248     ]
249     [=
250         |'abstraction':
251             oPushIndentPos(0)                               203
252             oSetupPos(0)                                    246
253             oPushCurrentTokenMark                           237
254             'function'
255             oSetupGcurPosNc                                 217
256             oPushNextTokenMark                              239
```

```
257             oPushIndentPos(0)                                   203
258             @FunctionDeclaration                                  *
259             oCcompress(0)                                       242
260             oPopPosAndMark                                      243
261             ';'
262             oMoveCmnts                                          245
263             oSetupPos(0)                                        252
264             'invariant'
265             oPushIndentPos(0)                                   257
266             oSetupPos(0)                                        263
267             oPushCurrentTokenMark                               253
268             '('
269             oPushCurrentTokenMark                               267
270             oPushCurrentTokenMark                               269
271             oPushGcurPosNc                                      130
272             oSetupGcurPosNc                                     255
273             oPushGcurPosNc                                      271
274             @Expression                                           *
275             ')'
276             oSetupPos(0)                                        266
277             oCcompress(0)                                       259
278             oPopPosAndMark                                      260
279             oCcompress(0)                                       277
280             oPopPosAndMark                                      278
281             ';'
282             oMoveCmnts                                          262
283             oSetupPos(0)                                        276
284         |'invariant':
285             oPushIndentPos(0)                                   265
286             oSetupPos(0)                                        283
287             oPushCurrentTokenMark                               270
288             '('
289             oPushCurrentTokenMark                               287
290             oPushCurrentTokenMark                               289
291             oPushGcurPosNc                                      273
292             oSetupGcurPosNc                                     272
293             oPushGcurPosNc                                      291
294             @Expression                                         274
295             ')'
296             oSetupPos(0)                                        286
297             oCcompress(0)                                       279
298             oPopPosAndMark                                      280
299             oCcompress(0)                                       297
300             oPopPosAndMark                                      298
301             ';'
302             oMoveCmnts                                          282
303             oSetupPos(0)                                        296
304         |*:
305     ]
306     [=
307         |'finally':
308             oPushIndentPos(0)                                   265
309             oPushCurrentTokenMark                               290
310             oSetupPos(0)                                        303
311             oPushNextTokenMark                                  256
312             oPushIndentPos(0)                                   265
```

```
    313              @RoutineDefinition                                  241
    314              oCcompress(0)                                       299
    315              oPopPosAndMark                                      300
    316              ';'
    317              oMoveCmnts                                          302
    318              oSetupPos(0)                                        310
    319          |*:
    320      ]
    321      oCcompress(0)                                               314
    322      oPopMark                                                    163
    323      oPopPosAndMark;                                             315
    324
    325
    326
    327 ImportClause :                                                    *
    328
    329     '('
    330     oPushCurrentTokenMark                                        309
    331     oSetupPos(0)                                                 318
    332     oCcompress(0)                                                314
    333     oPushCurrentTokenMark                                        309
    334     oPushGcurPosNc                                               293
    335     @ImportItem                                                   *
    336     {[=
c   337          |',':
c   338             oSetupPos(0)
c   339             oPushGcurPosNc
c   340             oPushCurrentTokenMark
c   341             @ImportItem
c   342          |')':
c   343             oSetupPos(0)
c   344             oCcompress(0)
c   345             oPopMark
c   346             >
c   347     ]}
    348     [=
    349         |'thus':
    350            oSetupPos(0)                                          318
    351            '('
    352            oPushIndentPos(0)                                     312
    353            oPushCurrentTokenMark                                 333
    354            oPushCopyPos(0)                                       240
    355            oSetupPos(0)                                          318
    356            oPushCurrentTokenMark                                 353
    357            @ThusItem                                              *
    358            {[=
c   359                 |',':
c   360                    oSetupPos(0)
c   361                    oPushGcurPosNc
c   362                    oPushCurrentTokenMark
c   363                    @ThusItem
c   364                 |')':
c   365                    oSetupPos(0)
c   366                    oCcompress(0)
c   367                    oPopPosAndMark
c   368                    >
```

```
c  369            ])
   370          |*:
   371      ]
   372      oPopPosAndMark;                                          323
   373
   374
   375
   376  ImportItem :                                                 335
   377
   378      oPushCopyPos(0)                                          354
   379      oPushCopyMark(0)                                         155
   380      @Pervasive                                               156
   381      oPushCurrentTokenMark                                    356
   382      oPushGcurPosNc                                           334
   383      oPushNextTokenMark                                       311
   384      @BindingCondition                                          *
   385      'ident'
   386      oSetupGcurPosNc                                          292
   387      oCcompress(0)                                            332
   388      oPopMark                                                 322
   389      oPopPosAndMark;                                          323
   390
   391
   392
   393  Pervasive :                                                  380
   394
   395      [=
   396          |'pervasive':
   397              oSetupPos(0)                                     355
   398              oCcompress(0)                                    387
   399          |*:
   400      ]
   401      oPopPosAndMark;                                          389
   402
   403
   404
   405  BindingCondition :                                           384
   406
   407      oPushCurrentTokenMark                                    381
   408      [=
   409          |'var':
   410             oSetupPos(0)                                      355
   411          |'readonly':
   412             oSetupPos(0)                                      355
   413          |'const':
   414             oSetupPos(0)                                      355
   415          |*:
   416      ]
   417      oCcompress(0)                                            398
   418      oPopMark                                                 388
   419      oPopPosAndMark;                                          389
   420
   421
   422
   423  ThusItem :                                                   357
   424      oPushCurrentTokenMark                                    381
```

```
    425      [=
    426          |'pervasive':
    427             oSetupPos(0)                                    355
    428             oCcompress(0)                                   398
    429          |*:
    430      ]
    431      [=
    432          |'var':
    433             oSetupGcurPosNc                                 386
    434          |'readonly':
    435             oSetupGcurPosNc                                 386
    436          |'const':
    437             oSetupGcurPosNc                                 386
    438          |*:
    439      ]
    440      'Ident'
    441      oPushIndentPos(0)                                       352
    442      OSetupPos(0)                                              *
    443      oCcompress(0)                                           428
    444      oPopPosAndMark                                          419
    445      oPopPosAndMark;                                         444
    446
    447
    448
    449  ExportClause :                                              204
    450
    451      '('
    452      oPushCurrentTokenMark                                   381
    453      oSetupPos(0)                                            355
    454      oCcompress(0)                                           443
    455      oPushGcurPosNc                                          382
    456      oPushCurrentTokenMark                                   381
    457      ∂ExportItem                                               *
    458      {[=
c   459           |')':
c   460              oSetupPos(0)
c   461              oCcompress(0)
c   462              oPopMark
c   463              >
c   464           |',':
c   465              oSetupPos(0)
c   466              oPushGcurPosNc
c   467              oPushCurrentTokenMark
c   468              ∂ExportItem
c   469      ]}
    470      oPopPosAndMark;                                         445
    471
    472
    473
    474  ExportItem :                                                457
    475
    476      [=
    477          |':=':
    478             oSetupPos(0)                                    355
    479
    480             oCcompress(0)                                   454
```

```
481         |'-':
482             oSetupPos(0)                                355
483             oCcompress(0)                               480
484         |'^':
485             oSetupPos(0)                                355
486             oCcompress(0)                               483
487         |'Ident':
488             oPushCurrentTokenMark                       456
489             oSetupPos(0)                                355
490             oCcompress(0)                               486
491             [-
492                 |'..':
493                     oSetupPos(0)                        355
494                     'Ident'
495                     oSetupGcurPosNc                     386
496
497
498                 |*:
499                     oPushIndentPos(0)                   352
500                     oPushNextTokenMark                  383
501                     ∂WithClause                           *
502             ]
503             oCcompress(0)                               490
504             oPopMark                                    418
505         |*:
506             oPushCurrentTokenMark                       456
507             oPushCopyPos(0)                             354
508             oPushCopyMark(0)                            379
509             ∂BindingCondition                           405
510             'Ident'
511             oPushIndentPos(0)                           352
512             oSetupPos(0)                                355
513             oPushIndentPos(0)                           511
514             oPushNextTokenMark                          500
515             ∂WithClause                                 501
516             oCcompress(0)                               503
517             oPopPosAndMark                              470
518     ]
519     oPopPosAndMark;                                     517
520
521
522
523
524
525
526
527 WithClause :                                           515
528     oPushCurrentTokenMark                               456
529     [-
530         |'with':
531             oSetupPos(0)                                512
532             oCcompress(0)                               516
533             oPushIndentPos(0)                           513
534             oPushCurrentTokenMark                       456
535             oPushNextTokenMark                          514
536             ∂ExportClause                               449
```

```
   537              oCcompress(0)                                          532
   538              oPopMark                                               504
   539          |*:
   540      ]
   541      oPopMark                                                       538
   542      oPopPosAndMark;                                                519
   543
   544
   545
   546 Declarations :                                                      233
   547
   548      (
c  549      [=
c  550           |'var':
c  551              oPushIndentPos(0)
c  552              oPushCurrentTokenMark
c  553              oPushCurrentTokenMark
c  554              oSetupPos(0)
c  555              oCcompress(0)
c  556              oPushIndentPos(0)
c  557              ∂VariableDeclaration
c  558              oCcompress(0)
c  559              oPopPosAndMark
c  560           |'bind':
c  561              oPushIndentPos(0)
c  562              oPushCurrentTokenMark
c  563              oPushCurrentTokenMark
c  564              oSetupPos(0)
c  565              oCcompress(0)
c  566              oPushIndentPos(0)
c  567              ∂VariableBinding
c  568              oCcompress(0)
c  569              oPopPosAndMark
c  570           |'assert':
c  571              oPushIndentPos(0)
c  572              oPushCurrentTokenMark
c  573              oPushCurrentTokenMark
c  574              oSetupPos(0)
c  575              oCcompress(0)
c  576              oPushIndentPos(0)
c  577              ∂AssertStatement
c  578              oCcompress(0)
c  579              oPopPosAndMark
c  580           |'pervasive':
c  581              oPushIndentPos(0)
c  582              oPushCurrentTokenMark
c  583              oPushCurrentTokenMark
c  584              oSetupPos(0)
c  585              oCcompress(0)
c  586              [=
c  587                   |'const':
c  588                      oSetupGcurPosNc
c  589                      oCcompress(0)
c  590                      oPushIndentPos(0)
c  591                      ∂ConstantDeclaration
c  592                   |'type':
```

```
c 593                       oSetupGcurPosNc
c 594                       oCcompress(0)
c 595                       oPushIndentPos(0)
c 596                       aTypeDeclaration
c 597                   |'converter':
c 598                       oSetupGcurPosNc
c 599                       oCcompress(0)
c 600                       oPushIndentPos(0)
c 601                       aConverterDeclaration
c 602                   |'inline':
c 603                       oSetupGcurPosNc
c 604                       oCcompress(0)
c 605                       oPushIndentPos(0)
c 606                       aInlineProcedureDeclaration
c 607                   |'procedure':
c 608                       oSetupGcurPosNc
c 609                       oCcompress(0)
c 610                       oPushIndentPos(0)
c 611                       aProcedureDeclaration
c 612                   |'function':
c 613                       oSetupGcurPosNc
c 614                       oCcompress(0)
c 615                       oPushIndentPos(0)
c 616                       aFunctionDeclaration
c 617               ]
c 618               oCcompress(0)
c 619               oPopPosAndMark
c 620           |'const':
c 621               oPushIndentPos(0)
c 622               oPushCurrentTokenMark
c 623               oPushCurrentTokenMark
c 624               oSetupPos(0)
c 625               oCcompress(0)
c 626               oPushIndentPos(0)
c 627               aConstantDeclaration
c 628               oCcompress(0)
c 629               oPopPosAndMark
c 630           |'type':
c 631               oPushIndentPos(0)
c 632               oPushCurrentTokenMark
c 633               oPushCurrentTokenMark
c 634               oSetupPos(0)
c 635               oCcompress(0)
c 636               oPushIndentPos(0)
c 637               aTypeDeclaration
c 638               oCcompress(0)
c 639               oPopPosAndMark
c 640           |'converter':
c 641               oPushIndentPos(0)
c 642               oPushCurrentTokenMark
c 643               oPushCurrentTokenMark
c 644               oSetupPos(0)
c 645               oCcompress(0)
c 646               oPushIndentPos(0)
c 647               aConverterDeclaration
c 648               oCcompress(0)
```

```
c  649              oPopPosAndMark
c  650           |'Inline':
c  651              oPushIndentPos(0)
c  652              oPushCurrentTokenMark
c  653              oPushCurrentTokenMark
c  654              oSetupPos(0)
c  655              oCcompress(0)
c  656              oPushIndentPos(0)
c  657              @InlineProcedureDeclaration
c  658              oCcompress(0)
c  659              oPopPosAndMark
c  660           |'procedure':
c  661              oPushIndentPos(0)
c  662              oPushCurrentTokenMark
c  663              oPushCurrentTokenMark
c  664              oSetupPos(0)
c  665              oCcompress(0)
c  666              oPushCopyPos(0)
c  667              @ProcedureDeclaration
c  668              oCcompress(0)
c  669              oPopPosAndMark
c  670
c  671
c  672           |'function':
c  673              oPushIndentPos(0)
c  674              oPushCurrentTokenMark
c  675              oPushCurrentTokenMark
c  676              oSetupPos(0)
c  677              oCcompress(0)
c  678              oPushCopyPos(0)
c  679              @FunctionDeclaration
c  680              oCcompress(0)
c  681              oPopPosAndMark
c  682           |';':
c  683              oMoveCmnts
c  684              oSetupPos(0)
c  685           |*:
c  686              >
c  687        ]
c  688        ';'
c  689        oMoveCmnts
c  690        oSetupPos(0)
c  691     }
   692     oPopPosAndMark;                                          519
   693
   694
   695
   696 VariableDeclaration :                                        *
   697
   698
   699     'Ident'
   700     oSetupPos(0)                                             512
   701     oCcompress(0)                                            537
   702     oPushCurrentTokenMark                                    534
   703     [=
   704         |'(':
```

```
705             oPushGcurPosNc                                  455
706             oSetupGcurPosNc                                 495
707             'at'
708             oPushCurrentTokenMark                           702
709             oSetupGcurPosNc                                 495
710             oPushGcurPosNc                                  455
711             ∂Expression                                     294
712             ')'
713             oSetupPos(0)                                    512
714             oPopPos                                         226
715         {',':
716             oSetupPos(0)                                    512
717             oPushCopyPos(0)                                 354
718             oPushCopyMark(0)                                508
719             ∂IdentifierList                                   *
720         }*:
721     ]
722     oCcompress(0)                                           701
723     oPopMark                                                538
724     oCcompress(0)                                           701
725     ':'
726     oPushCurrentTokenMark                                   702
727     oSetupPos(0)                                            512
728     oPushGcurPosNc                                          455
729     oPushNextTokenMark                                      535
730     ∂TypeDefinition                                           *
731     oCcompress(0)                                           724
732     [=
733         {':=':
734             oPushCurrentTokenMark                           726
735             oSetupPos(0)                                    727
736             oPushGcurPosNc                                  728
737             oPushNextTokenMark                              729
738             ∂Expression                                     294
739             oCcompress(0)                                   731
740             oPopMark                                        723
741             oCcompress(0)                                   739
742         }*:
743     ]
744     oPopMark                                                740
745     oPopPosAndMark;                                         519
746
747
748
749 VariableBinding :                                             *
750
751     [=
752         {'(':
753             oPushCopyPos(0)                                 717
754             oPushCurrentTokenMark                           726
755             oPushCurrentTokenMark                           754
756             oSetupPos(0)                                    727
757             oPushGcurPosNc                                  736
758             ∂RenameVariable                                   *
759             ([=
c 760                 {')':
```

```
c  761                    oSetupPos(0)
c  762                    oCcompress(0)
c  763                    oPopPosAndMark
c  764                    >
c  765                 |',':
c  766                    oPushCurrentTokenMark
c  767                    oPushCurrentTokenMark
c  768                    oSetupPos(0)
c  769                    oPushGcurPosNc
c  770                    @RenameVariable
c  771                    oCcompress(0)
c  772                    oPopMark
c  773            ]}
   774         |*:
   775            oPushCopyPos(0)                                    717
   776            oPushCopyMark(0)                                   718
   777            @RenameVariable                                    758
   778     ]
   779     oPopPosAndMark;                                           519
   780
   781
   782
   783 RenameVariable :                                             777
   784
   785     oPushCopyPos(0)                                           717
   786     oPushCopyMark(0)                                          776
   787     @VarBindingCondition                                        *
   788     'Ident'
   789     oPushIndentPos(0)                                         533
   790     oSetupPos(0)                                              756
   791     oCcompress(0)                                             741
   792     'to'
   793     oSetupGcurPosNc                                           495
   794     'Ident'
   795     oSetupGcurPosNc                                           793
   796     oPushIndentPos(0)                                         789
   797     oPushNextTokenMark                                        737
   798     @Variable                                                   *
   799     oCcompress(0)                                             791
   800     oPopPos                                                   714
   801     oPopPosAndMark;                                           779
   802
   803
   804
   805 VarBindingCondition :                                        787
   806
   807     [=
   808         |'var':
   809            oSetupPos(0)                                       790
   810         |'readonly':
   811            oSetupPos(0)                                       790
   812         |*:
   813     ]
   814     oCcompress(0)                                             799
   815     oPopPosAndMark;                                           801
   816
```

```
   817
   818
   819 ConstantDeclaration :                                                  *
   820
   821     oPushCopyPos(0)                                                    717
   822     oPushCopyMark(0)                                                   786
   823     ∂IdentifierList                                                    719
   824     oCcompress(0)                                                      799
   825     oPushCurrentTokenMark                                              755
   826     [=
   827         |':=':
   828            oSetupPos(0)                                                790
   829            oPushGcurPosNc                                              757
   830            oPushNextTokenMark                                          797
   831            ∂Expression                                                 738
   832         |':':
   833            oSetupPos(0)                                                790
   834            oPushGcurPosNc                                              829
   835            oPushNextTokenMark                                          830
   836            ∂TypeDefinition                                             730
   837            oCcompress(0)                                               824
   838            ':='
   839            oPushCurrentTokenMark                                       825
   840            oSetupPos(0)                                                790
   841            [=
   842                |'(':
   843                   oPushGcurPosNc                                       834
   844                   oPushGcurPosNc                                       834
   845                   oPushCurrentTokenMark                                839
   846                   oPushCurrentTokenMark                                839
   847                   oSetupGcurPosNc                                      795
   848                   ∂StructureConstant                                    *
   849                   ')'
   850                   oSetupPos(0)                                         840
   851                   oCcompress(0)                                        837
   852                   oPopPosAndMark                                       815
   853                |*:
   854                   oPushGcurPosNc                                       834
   855                   oPushNextTokenMark                                   835
   856                   ∂Expression                                          831
   857            ]
   858            oCcompress(0)                                               851
   859            oPopMark                                                    740
   860     ]
   861     oCcompress(0)                                                      851
   862     oPopMark                                                           859
   863     oPopPosAndMark;                                                    852
   864
   865
   866
   867 StructureConstant :                                                   848
   868
   869     ([=
c  870          |'(':
c  871             oPushGcurPosNc
c  872             oPushGcurPosNc
```

```
c  873              oPushCurrentTokenMark
c  874              oPushCopyMark(0)
c  875              oSetupGcurPosNc
c  876              @StructureConstant
c  877              '}'
c  878              oSetupPos(0)
c  879              oCcompress(0)
c  880              oPopPosAndMark
c  881           |*:
c  882              oPushGcurPosNc
c  883              oPushNextTokenMark
c  884              @Expression
c  885       ]
c  886       [-
c  887           |',':
c  888              oSetupPos(0)
c  889           |*:
c  890              >
c  891       ]
c  892     }
   893     oPopPosAndMark;                                          863
   894
   895
   896
   897 ConverterDeclaration :                                       *
   898
   899     'Ident'
   900     oPushCurrentTokenMark                                    839
   901     oSetupPos(0)                                             840
   902     oCcompress(0)                                            861
   903     '('
   904     oPushCurrentTokenMark                                    839
   905     oPushIndentPos(0)                                        796
   906     oSetupPos(0)                                             840
   907     [-
   908         |'Ident':
   909            oSetupGcurPosNc                                   795
   910         |'procedure':
   911            oSetupGcurPosNc                                   795
   912         |'function':
   913            oSetupGcurPosNc                                   795
   914     ]
   915     ')'
   916     oSetupPos(0)                                             906
   917     oCcompress(0)                                            902
   918     oPopPosAndMark                                           863
   919     'returns'
   920     oSetupPos(0)                                             906
   921     'Ident'
   922     oSetupGcurPosNc                                          795
   923     oCcompress(0)                                            917
   924     oPopMark                                                 862
   925     oPopPosAndMark;                                          918
   926
   927
   928
```

```
929 TypeDeclaration :                                                   161
930
931     'Ident'
932     oSetupPos(0)                                                    920
933     oCcompress(0)                                                   923
934     [=
935         |'(':
936             oPushCurrentTokenMark                                   904
937             oPushCurrentTokenMark                                   904
938             oPushIndentPos(0)                                       905
939             oSetupPos(0)                                            920
940             oPushCopyPos(0)                                         717
941             ∂TypeFormalList                                           *
942             ')'
943             oSetupPos(0)                                            939
944             oCcompress(0)                                           933
945             oPopPosAndMark                                          925
946         |*:
947     ]
948     '='
949     oPushCurrentTokenMark                                           904
950     oSetupPos(0)                                                    943
951     [=
952         |'pre':
953             oSetupGcurPosNc                                         922
954             [=
955                 |'(':
956                     oPushGcurPosNc                                  834
957                     oPushCurrentTokenMark                           904
958                     oPushCurrentTokenMark                           904
959                     oSetupGcurPosNc                                 922
960                     oPushGcurPosNc                                  834
961                     ∂Expression                                     831
962                     ')'
963                     oSetupPos(0)                                    943
964                     oCcompress(0)                                   944
965                     oPopPosAndMark                                  945
966                 |*:
967             ]
968             oCcompress(0)                                           964
969             oPopMark                                                924
970             ';'
971             oMoveCmnts                                              317
972             oSetupPos(0)                                            943
973         |*:
974     ]
975     [=
976         |'forward':
977             oPushIndentPos(0)                                       905
978             oSetupPos(0)                                            972
979             oPopPos                                                 800
980         |*:
981             oPushIndentPos(0)                                       905
982             oPushCurrentTokenMark                                   904
983             ∂TypeDefinition                                         836
984     ]
```

```
   985      oCcompress(0)                                                  964
   986      oPopMark                                                       969
   987      oPopPosAndMark;                                                965
   988
   989
   990
   991  IdentifierList :                                                   823
   992
   993      'Ident'
   994      oPushCurrentTokenMark                                          982
   995      oSetupPos(0)                                                   978
   996      oCcompress(0)                                                  985
   997      {[=
c  998           |',':
c  999              oSetupPos(0)
c 1000              'Ident'
c 1001              oSetupGcurPosNc
c 1002           |*:
c 1003              >
c 1004      ])
  1005      oCcompress(0)                                                  996
  1006      oPopMark                                                       986
  1007      oPopPosAndMark;                                                987
  1008
  1009
  1010
  1011  TypeFormalList :                                                   941
  1012
  1013      oPushCopyMark(0)                                               822
  1014      oPushIndentPos(0)                                              905
  1015      ∂Pervasive                                                     393
  1016      [=
  1017          |'const':
  1018             oSetupGcurPosNc                                         922
  1019             oCcompress(0)                                          1005
  1020          |*:
  1021      ]
  1022      oPushIndentPos(0)                                             1014
  1023      oPushCopyMark(0)                                               822
  1024      ∂IdentifierList                                                991
  1025      oCcompress(0)                                                 1005
  1026      ':'
  1027      oPushCurrentTokenMark                                          982
  1028      oPushIndentPos(0)                                             1014
  1029      oSetupPos(0)                                                   978
  1030      oPushGcurPosNc                                                 834
  1031      oPushNextTokenMark                                             835
  1032      ∂IndexType                                                       *
  1033      oCcompress(0)                                                 1025
  1034      oPopPosAndMark                                                1007
  1035      oCcompress(0)                                                 1033
  1036      {[=
c 1037           |',':
c 1038              oPushCurrentTokenMark
c 1039              oPushCurrentTokenMark
c 1040              oSetupPos(0)
```

```
c 1041              oPushGcurPosNc
c 1042              aTypeFormalList
c 1043              oCcompress(0)
c 1044              oPopMark
c 1045           |*:
c 1046              >
c 1047      ]}
  1048      oPopPosAndMark;                                          1034
  1049
  1050
  1051
  1052  InlineProcedureDeclaration :                                  *
  1053
  1054      [=
  1055          |'procedure':
  1056              oPushCurrentTokenMark                            1027
  1057              oSetupPos(0)                                     1029
  1058              oCcompress(0)                                    1035
  1059              oPushIndentPos(0)                                1028
  1060              oPushCurrentTokenMark                            1027
  1061              aProcedureDeclaration                             *
  1062          |'function':
  1063              oPushCurrentTokenMark                            1060
  1064              oSetupPos(0)                                     1029
  1065              oCcompress(0)                                    1058
  1066              oPushIndentPos(0)                                1059
  1067              oPushCurrentTokenMark                            1060
  1068              aFunctionDeclaration                              258
  1069      ]
  1070      oCcompress(0)                                            1065
  1071      oPopMark                                                 1006
  1072      oPopPosAndMark;                                          1048
  1073
  1074
  1075
  1076  TypeDefinition :                                              983
  1077
  1078      [=
  1079          |'(':
  1080              oPushCopyPos(0)                                   940
  1081              oPushCurrentTokenMark                            1067
  1082              oPushCopyMark(0)                                 1023
  1083              oSetupPos(0)                                     1029
  1084              oCcompress(0)                                    1070
  1085              aEnumeratedTypeDefinition                         *
  1086          |'array':
  1087              oPushCurrentTokenMark                            1081
  1088              oSetupPos(0)                                     1083
  1089              oCcompress(0)                                    1084
  1090              oPushGcurPosNc                                   1030
  1091              oPushCurrentTokenMark                            1081
  1092              aIndexType                                       1032
  1093              'of'
  1094              oSetupPos(0)                                     1083
  1095              oPushGcurPosNc                                   1090
  1096              oPushNextTokenMark                               1031
```

```
1097            ∂TypeDefinition                                  1076
1098         ]'record':
1099            oPushCopyPos(0)                                   940
1100            oPushCurrentTokenMark                            1091
1101            oSetupPos(0)                                     1094
1102            oCcompress(0)                                    1089
1103            oPushCopyMark(0)                                 1082
1104            ∂RecordBody                                         *
1105            oPushCopyPos(0)                                   940
1106            oPushNextTokenMark                               1096
1107            ∂RecordTrailer                                      *
1108         ]'module':
1109            oPushCopyPos(0)                                  1105
1110            oPushCurrentTokenMark                            1100
1111            oSetupPos(0)                                     1101
1112            oCcompress(0)                                    1102
1113            oPushNextTokenMark                               1106
1114            ∂ModuleBody                                       173
1115            oPushCopyPos(0)                                  1105
1116            oPushNextTokenMark                               1113
1117            ∂ModuleTrailer                                      *
1118         ]'set':
1119            oPushCurrentTokenMark                            1110
1120            oSetupPos(0)                                     1111
1121            oCcompress(0)                                    1112
1122            oPushIndentPos(0)                                1066
1123            'of'
1124            oSetupGcurPosNc                                   922
1125            oPushNextTokenMark                               1116
1126            ∂IndexType                                       1092
1127         ]'machine':
1128            oSetupPos(0)                                     1111
1129            'dependent'
1130            oSetupGcurPosNc                                  1124
1131            oCcompress(0)                                    1121
1132            [=
1133                ]'record':
1134                   oPushIndentPos(0)                         1122
1135                   oPushCurrentTokenMark                     1110
1136                   oSetupPos(0)                              1111
1137                   [=
1138                       ]'aligned':
1139                          oPushCurrentTokenMark              1135
1140                          oPushIndentPos(0)                  1122
1141                          oSetupPos(0)                       1136
1142                          'mod'
1143                          oPushIndentPos(0)                  1122
1144                          oSetupGcurPosNc                    1130
1145                          oPushNextTokenMark                 1125
1146                          ∂Expression                         831
1147                          oCcompress(0)                      1131
1148                          oPopPosAndMark                     1072
1149                       ]*:
1150                   ]
1151                   oPushCopyPos(0)                           1115
1152                   oPushCopyPos(0)                           1115
```

```
1153                    oPushNextTokenMark                          1145
1154                    @MDRecordBody                                 *
1155                    oPushNextTokenMark                          1153
1156                    @RecordTrailer                              1107
1157                    oPopPos                                      979
1158                 |'module':
1159                    oPushIndentPos(0)                           1143
1160                    oPushCurrentTokenMark                       1135
1161                    oSetupPos(0)                                1141
1162                    oPushCopyPos(0)                             1152
1163                    oPushNextTokenMark                          1155
1164                    @ModuleBody                                 1114
1165                    oPushCopyPos(0)                             1162
1166                    oPushNextTokenMark                          1163
1167                    @ModuleTrailer                              1117
1168                    oPopPos                                     1157
1169              ]
1170           |'collection':
1171              oPushCurrentTokenMark                            1160
1172              oSetupPos(0)                                     1161
1173              oCcompress(0)                                    1147
1174              'of'
1175              oSetupGcurPosNc                                  1144
1176              oPushIndentPos(0)                                1143
1177              oPushNextTokenMark                               1166
1178              @CollectionTypeDefinition                          *
1179           |'counted':
1180              oPushCurrentTokenMark                            1160
1181              oSetupPos(0)                                     1161
1182              oCcompress(0)                                    1173
1183              [=
1184                 |'collection':
1185                    oSetupGcurPosNc                            1175
1186                    'of'
1187                    oSetupGcurPosNc                            1175
1188                    oPushIndentPos(0)                          1176
1189                    oPushNextTokenMark                         1177
1190                    @CollectionTypeDefinition                  1178
1191                 |*:
1192                    oPushIndentPos(0)                          1176
1193                    oPushNextTokenMark                         1189
1194                    @Expression                                1146
1195                    oCcompress(0)                              1182
1196                    'collection'
1197                    oPushCurrentTokenMark                      1160
1198                    oSetupPos(0)                               1161
1199                    'of'
1200                    oSetupGcurPosNc                            1175
1201                    oPushIndentPos(0)                          1176
1202                    oPushNextTokenMark                         1193
1203                    @CollectionTypeDefinition                  1190
1204                    oCcompress(0)                              1195
1205                    oPopMark                                   1071
1206              ]
1207           |'checkable':
1208              oPushCurrentTokenMark                            1197
```

```
1209            oSetupPos(0)                                    1198
1210            oCcompress(0)                                   1204
1211            'collection'
1212            oPushIndentPos(0)                               1201
1213            oPushCurrentTokenMark                           1197
1214            oSetupPos(0)                                    1198
1215            'of'
1216            oSetupGcurPosNc                                 1200
1217            oPushIndentPos(0)                               1212
1218            oPushNextTokenMark                              1202
1219            ∂CollectionTypeDefinition                       1203
1220            oCcompress(0)                                   1210
1221            oPopPosAndMark                                  1148
1222        |'^':
1223            oPushCurrentTokenMark                           1213
1224            oSetupPos(0)                                    1214
1225            oCcompress(0)                                   1220
1226            'Ident'
1227            oPushIndentPos(0)                               1217
1228            oSetupPos(0)                                    1214
1229            oPushIndentPos(0)                               1227
1230            oPushNextTokenMark                              1218
1231            ∂Variable                                        798
1232            oCcompress(0)                                   1225
1233            oPopPos                                         1168
1234        |'packed':
1235            oPushCurrentTokenMark                           1213
1236            oSetupPos(0)                                    1228
1237            oCcompress(0)                                   1232
1238            (=
1239                |'array':
1240                    oSetupGcurPosNc                         1216
1241                    oPushIndentPos(0)                       1229
1242                    oPushNextTokenMark                      1230
1243                    ∂IndexType                              1126
1244                    oCcompress(0)                           1237
1245                    'of'
1246                    oPushCurrentTokenMark                   1213
1247                    oSetupPos(0)                            1228
1248                    oPushGcurPosNc                          1095
1249                    oPushNextTokenMark                      1242
1250                    ∂TypeDefinition                         1097
1251                    oCcompress(0)                           1244
1252                    oPopMark                                1205
1253                |'record':
1254                    oSetupGcurPosNc                         1216
1255                    oPushCopyPos(0)                         1165
1256                    oPushNextTokenMark                      1249
1257                    ∂RecordBody                             1104
1258                    oPushCopyPos(0)                         1165
1259                    oPushNextTokenMark                      1256
1260                    ∂RecordTrailer                          1156
1261                |'module':
1262                    oSetupGcurPosNc                         1216
1263                    oPushCopyPos(0)                         1258
1264                    oPushNextTokenMark                      1259
```

```
1265                    @ModuleBody                                     1164
1266                    oPushCopyPos(0)                                 1258
1267                    oPushNextTokenMark                              1264
1268                    @ModuleTrailer                                  1167
1269                |'set':
1270                    oSetupGcurPosNc                                 1216
1271                    'of'
1272                    oPushCurrentTokenMark                           1246
1273                    oSetupPos(0)                                    1247
1274                    oPushGcurPosNc                                  1248
1275                    oPushNextTokenMark                              1267
1276                    @IndexType                                      1243
1277                    oCcompress(0)                                   1251
1278                    oPopMark                                        1252
1279                |'machine':
1280                    oSetupGcurPosNc                                 1216
1281                    'dependent'
1282                    oSetupGcurPosNc                                 1216
1283                    'module'
1284                    oSetupGcurPosNc                                 1216
1285                    oPushCopyPos(0)                                 1266
1286                    oPushNextTokenMark                              1275
1287                    @ModuleBody                                     1265
1288                    oPushCopyPos(0)                                 1266
1289                    oPushNextTokenMark                              1286
1290                    @ModuleTrailer                                  1268
1291            ]
1292        |*:
1293            oPushCurrentTokenMark                                   1246
1294            oPushCopyPos(0)                                         1288
1295            oPushCopyMark(0)                                        1103
1296            @NameOrRangeTypeDefinition                                *
1297    ]
1298    oCcompress(0)                                                   1277
1299    oPopMark                                                        1278
1300    oPopPosAndMark;                                                 1221
1301
1302
1303
1304 IndexType :                                                        1276
1305
1306    [=
1307        |'(':
1308            oPushCopyPos(0)                                         1288
1309            oPushCurrentTokenMark                                   1246
1310            oPushCopyMark(0)                                        1295
1311            oSetupPos(0)                                            1247
1312            @EnumeratedTypeDefinition                               1085
1313        |*:
1314            oPushCurrentTokenMark                                   1309
1315            oPushCopyPos(0)                                         1288
1316            oPushCopyMark(0)                                        1310
1317            @NameOrRangeTypeDefinition                              1296
1318    ]
1319    oCcompress(0)                                                   1298
1320    oPopMark                                                        1299
```

```
1321      oPopPosAndMark;                                                    1300
1322
1323
1324
1325  EnumeratedTypeDefinition :                                             1312
1326
1327      oPushGcurPosNc                                                     1274
1328      oPushCopyMark(0)                                                   1316
1329      aIdentifierList                                                    1024
1330      oCcompress(0)                                                      1319
1331      ')'
1332      oSetupPos(0)                                                       1311
1333      oCcompress(0)                                                      1330
1334      oPopPosAndMark;                                                    1321
1335
1336
1337
1338  NameOrRangeTypeDefinition :                                            1317
1339
1340      [=
1341          |'Ident':
1342             oPushCurrentTokenMark                                       1309
1343             oSetupPos(0)                                                1332
1344             oCcompress(0)                                               1333
1345             oPushIndentPos(0)                                           1229
1346             oPushNextTokenMark                                          1289
1347             aVariable                                                   1231
1348             oCcompress(0)                                               1344
1349             [=
1350                 |'..':
1351                    oPushCurrentTokenMark                                1309
1352                    oSetupPos(0)                                         1332
1353                 |*:
1354                    [=
1355                        |'+':
1356                           oPushCurrentTokenMark                         1309
1357                           oPushCurrentTokenMark                         1309
1358                           oSetupPos(0)                                  1332
1359                           oPushGcurPosNc                                1274
1360                           aTerm                                           *
1361                           oCcompress(0)                                 1348
1362                           oPushCopyPos(0)                               1288
1363                           oPushNextTokenMark                            1346
1364                           aPartialSum                                     *
1365                           oCcompress(0)                                 1361
1366                           oPopMark                                      1320
1367                        |'-':
1368                           oPushCurrentTokenMark                         1309
1369                           oPushCurrentTokenMark                         1309
1370                           oSetupPos(0)                                  1332
1371                           oPushGcurPosNc                                1359
1372                           aTerm                                         1360
1373                           oCcompress(0)                                 1365
1374                           oPushCopyPos(0)                               1362
1375                           oPushNextTokenMark                            1363
1376                           aPartialSum                                   1364
```

```
1377                    oCcompress(0)                                   1373
1378                    oPopMark                                        1366
1379                '|'xor':
1380                    oPushCurrentTokenMark                           1309
1381                    oPushCurrentTokenMark                           1309
1382                    oSetupPos(0)                                    1332
1383                    oPushGcurPosNc                                  1371
1384                    @Term                                           1372
1385                    oCcompress(0)                                   1377
1386                    oPushCopyPos(0)                                 1374
1387                    oPushNextTokenMark                              1375
1388                    @PartialSum                                     1376
1389                    oCcompress(0)                                   1385
1390                    oPopMark                                        1378
1391                '|'*':
1392                    oPushCurrentTokenMark                           1309
1393                    oPushCurrentTokenMark                           1309
1394                    oSetupPos(0)                                    1332
1395                    oPushGcurPosNc                                  1383
1396                    @Factor                                           *
1397                    oCcompress(0)                                   1389
1398                    oPushCopyPos(0)                                 1386
1399                    oPushNextTokenMark                              1387
1400                    @PartialSum                                     1388
1401                    oCcompress(0)                                   1397
1402                    oPopMark                                        1390
1403                '|'div':
1404                    oPushCurrentTokenMark                           1309
1405                    oPushCurrentTokenMark                           1309
1406                    oSetupPos(0)                                    1332
1407                    oPushGcurPosNc                                  1395
1408                    @Factor                                         1396
1409                    oCcompress(0)                                   1401
1410                    oPushCopyPos(0)                                 1398
1411                    oPushNextTokenMark                              1399
1412                    @PartialSum                                     1400
1413                    oCcompress(0)                                   1409
1414                    oPopMark                                        1402
1415                '|'mod':
1416                    oPushCurrentTokenMark                           1309
1417                    oPushCurrentTokenMark                           1309
1418                    oSetupPos(0)                                    1332
1419                    oPushGcurPosNc                                  1407
1420                    @Factor                                         1408
1421                    oCcompress(0)                                   1413
1422                    oPushCopyPos(0)                                 1410
1423                    oPushNextTokenMark                              1411
1424                    @PartialSum                                     1412
1425                    oCcompress(0)                                   1421
1426                    oPopMark                                        1414
1427                '|':
1428                    oPopMark                                        1426
1429                    oPopPosAndMark                                  1334
1430                    >>
1431            ]
1432            '..'
```

```
1433                    oPushCurrentTokenMark                   1309
1434                    oSetupGcurPosNc                         1216
1435                ]
1436                oPushGcurPosNc                              1419
1437                oPushNextTokenMark                          1423
1438                @Sum                                          *
1439                oCcompress(0)                               1425
1440                oPopMark                                    1426
1441                oCcompress(0)                               1425
1442                oPopMark                                    1440
1443            |*:
1444                oPushCurrentTokenMark                       1309
1445                oPushCopyPos(0)                             1422
1446                oPushCopyMark(0)                            1316
1447                @Sum                                        1438
1448                '..'
1449                oPushCurrentTokenMark                       1309
1450                oSetupPos(0)                                1332
1451                oPushGcurPosNc                              1419
1452                oPushNextTokenMark                          1423
1453                @Sum                                        1447
1454                oCcompress(0)                               1441
1455                oPopMark                                    1442
1456                oCcompress(0)                               1454
1457                oPopMark                                    1455
1458    ]
1459    oPopPosAndMark;                                         1334
1460
1461
1462
1463 RecordBody :                                               1257
1464
1465    oPushCopyPos(0)                                         1422
1466    oPushCopyMark(0)                                        1446
1467    @DeclarationsInRecord                                     *
1468    [=
1469        |'case':
1470            oPushCurrentTokenMark                           1449
1471            oPushIndentPos(0)                               1345
1472            oSetupPos(0)                                    1450
1473            'Ident'
1474            oSetupGcurPosNc                                 1216
1475            [=
1476                |'default':
1477                    oPushIndentPos(0)                       1345
1478                    oPushCurrentTokenMark                   1449
1479                    oPushCurrentTokenMark                   1478
1480                    oSetupPos(0)                            1472
1481                    oPushGcurPosNc                          1451
1482                    @Expression                             1194
1483                    oCcompress(0)                           1456
1484                    oPopPosAndMark                          1459
1485                    oCcompress(0)                           1483
1486                |*:
1487            ]
1488            oPopMark                                        1457
```

```
  1489            'of'
  1490            oPushCurrentTokenMark                                   1479
  1491            oSetupPos(0)                                            1480
  1492            {[=
c 1493                 |'end':
c 1494                    oPopMark
c 1495                    oSetupPos(0)
c 1496                    'case'
c 1497                    oSetupGcurPosNc
c 1498                    >
c 1499                 |'otherwise':
c 1500                    oPushIndentPos(0)
c 1501                    oSetupPos(0)
c 1502                    oPopPos
c 1503                    '=>'
c 1504                    oPushCurrentTokenMark
c 1505                    oPushIndentPos(0)
c 1506                    oSetupPos(0)
c 1507                    oPushNextTokenMark
c 1508                    oPushGcurPosNc
c 1509                    ∂DeclarationsInRecord
c 1510                    oCcompress(0)
c 1511                    oPopPosAndMark
c 1512                    oCcompress(0)
c 1513                    oPopMark
c 1514                    'end'
c 1515                    oSetupPos(0)
c 1516                    'case'
c 1517                    oSetupGcurPosNc
c 1518                    >
c 1519                 |*:
c 1520                    oPushGcurPosNc
c 1521                    oPushNextTokenMark
c 1522                    ∂CaseLabel
c 1523                    {[=
c 1524                         |',':
c 1525                            oPushIndentPos(0)
c 1526                            oSetupPos(0)
c 1527                            oPushNextTokenMark
c 1528                            oPushGcurPosNc
c 1529                            ∂CaseLabel
c 1530                            oPopPos
c 1531                         |*:
c 1532                            oCcompress(0)
c 1533                            >
c 1534                    ]}
  1535                    '=>'
  1536                    oPushCurrentTokenMark                           1479
  1537                    oPushIndentPos(0)                               1345
  1538                    oSetupPos(0)                                    1480
  1539                    oPushGcurPosNc                                  1481
  1540                    oPushNextTokenMark                              1452
  1541                    ∂DeclarationsInRecord                           1467
  1542                    oCcompress(0)                                   1485
  1543                    'end'
  1544                    oSetupPos(0)                                    1538
```

```
1545                    oPopPos                                          1233
1546                    oPushGcurPosNc                                   1539
1547                    oPushNextTokenMark                               1540
1548                    @CaseLabel                                          *
1549                    oCcompress(0)                                    1542
1550                    oPopMark                                         1457
1551                    oCcompress(0)                                    1549
1552                    oPopMark                                         1550
1553                    ';'
1554                    oMoveCmnts                                        971
1555                    oPushCurrentTokenMark                            1536
1556                    oSetupPos(0)                                     1544
1557            ])
1558            oPopPos                                                  1545
1559            ';'
1560            oMoveCmnts                                               1554
1561            oSetupPos(0)                                             1556
1562        |*:
1563    ]
1564    oPopPosAndMark;                                                  1484
1565
1566
1567
1568 DeclarationsInRecord :                                              1541
1569
1570    ([=
c 1571          |'var':
c 1572             oPushCurrentTokenMark
c 1573             oPushCurrentTokenMark
c 1574             oPushIndentPos(0)
c 1575             oSetupPos(0)
c 1576             oCcompress(0)
c 1577             oPushIndentPos(0)
c 1578             @VariableDeclaration
c 1579             oCcompress(0)
c 1580             oPopPosAndMark
c 1581          |'const':
c 1582             oPushCurrentTokenMark
c 1583             oPushCurrentTokenMark
c 1584             oPushIndentPos(0)
c 1585             oSetupPos(0)
c 1586             oCcompress(0)
c 1587             oPushIndentPos(0)
c 1588             @ConstantDeclaration
c 1589             oCcompress(0)
c 1590             oPopPosAndMark
c 1591          |'pervasive':
c 1592             oPushCurrentTokenMark
c 1593             oPushCurrentTokenMark
c 1594             oPushIndentPos(0)
c 1595             oSetupPos(0)
c 1596             oCcompress(0)
c 1597             'const'
c 1598             oSetupGcurPosNc
c 1599             oPushIndentPos(0)
c 1600             @ConstantDeclaration
```

```
c 1601              oCcompress(0)
c 1602              oPopPosAndMark
c 1603           |'assert':
c 1604              oPushCurrentTokenMark
c 1605              oPushIndentPos(0)
c 1606              oSetupPos(0)
c 1607              oCcompress(0)
c 1608              oPushIndentPos(0)
c 1609              oPushNextTokenMark
c 1610              aAssertStatement
c 1611              oCcompress(0)
c 1612              oPopPosAndMark
c 1613           |';':
c 1614              oMoveCmnts
c 1615              oSetupPos(0)
c 1616           |*:
c 1617              >
c 1618       ]
c 1619       ';'
c 1620       oMoveCmnts
c 1621       oSetupPos(0)
c 1622      }
  1623      oPopPosAndMark;                                          1484
  1624
  1625
  1626
  1627 RecordTrailer :                                             1260
  1628
  1629      'end'
  1630      oSetupPos(0)                                             1561
  1631      oCcompress(0)                                            1551
  1632      [=
  1633          |'record':
  1634             oSetupGcurPosNc                                   1474
  1635          |'Ident':
  1636             oSetupGcurPosNc                                   1474
  1637      ]
  1638      oPopPosAndMark;                                          1484
  1639
  1640
  1641
  1642 ModuleTrailer :                                             1290
  1643
  1644      'end'
  1645      oSetupPos(0)                                             1561
  1646      oCcompress(0)                                            1631
  1647      [=
  1648          |'module':
  1649             oSetupGcurPosNc                                   1474
  1650          |'Ident':
  1651             oSetupGcurPosNc                                   1474
  1652      ]
  1653      oPopPosAndMark;                                          1484
  1654
  1655
  1656
```

```
  1657 MDRecordBody :                                                    1154
  1658
  1659     (
c 1660      [=
c 1661          {'var':
c 1662             oPushCurrentTokenMark
c 1663             oPushIndentPos(0)
c 1664             oSetupPos(0)
c 1665             oCcompress(0)
c 1666             'Ident'
c 1667             oSetupGcurPosNc
c 1668             '('
c 1669             oPushCurrentTokenMark
c 1670             oPushIndentPos(0)
c 1671             oSetupPos(0)
c 1672             'at'
c 1673             oPushCurrentTokenMark
c 1674             oSetupGcurPosNc
c 1675             oPushGcurPosNc
c 1676             ∂Expression
c 1677             [=
c 1678                 {'bits':
c 1679                    oPushCurrentTokenMark
c 1680                    oSetupPos(0)
c 1681                    oPushGcurPosNc
c 1682                    oPushNextTokenMark
c 1683                    ∂Sum
c 1684                    oCcompress(0)
c 1685                    '..'
c 1686                    oPushCurrentTokenMark
c 1687                    oSetupPos(0)
c 1688                    oPushGcurPosNc
c 1689                    oPushNextTokenMark
c 1690                    ∂Sum
c 1691                    oCcompress(0)
c 1692                    oPopMark
c 1693                    oCcompress(0)
c 1694                    oPopMark
c 1695                 }*:
c 1696             ]
c 1697             '}'
c 1698             oSetupPos(0)
c 1699             oCcompress(0)
c 1700             oPopPosAndMark
c 1701             ':'
c 1702             oPushCurrentTokenMark
c 1703             oSetupPos(0)
c 1704             oPushGcurPosNc
c 1705             oPushNextTokenMark
c 1706             ∂TypeDefinition
c 1707             oCcompress(0)
c 1708             [=
c 1709                 {':=':
c 1710                    oPushCurrentTokenMark
c 1711                    oPushCurrentTokenMark
c 1712                    oSetupPos(0)
```

```
c 1713                      oPushGcurPosNc
c 1714                      ∂Expression
c 1715                      oCcompress(0)
c 1716                      oPopMark
c 1717                      oCcompress(0)
c 1718                   |*:
c 1719               ]
c 1720               oPopPosAndMark
c 1721               oCcompress(0)
c 1722               oPopMark
c 1723            |'const':
c 1724               oPushCurrentTokenMark
c 1725               oPushCurrentTokenMark
c 1726               oPushIndentPos(0)
c 1727               oSetupPos(0)
c 1728               oCcompress(0)
c 1729               oPushIndentPos(0)
c 1730               ∂ConstantDeclaration
c 1731               oCcompress(0)
c 1732               oPopPosAndMark
c 1733            |'pervasive':
c 1734               oPushCurrentTokenMark
c 1735               oPushCurrentTokenMark
c 1736               oPushIndentPos(0)
c 1737               oSetupPos(0)
c 1738               oCcompress(0)
c 1739               'const'
c 1740               oSetupGcurPosNc
c 1741               oPushIndentPos(0)
c 1742               ∂ConstantDeclaration
c 1743               oCcompress(0)
c 1744               oPopPosAndMark
c 1745            |'assert':
c 1746               oPushCurrentTokenMark
c 1747               oPushIndentPos(0)
c 1748               oSetupPos(0)
c 1749               oCcompress(0)
c 1750               oPushIndentPos(0)
c 1751               oPushNextTokenMark
c 1752               ∂AssertStatement
c 1753               oPopPosAndMark
c 1754            |';':
c 1755               oMoveCmnts
c 1756               oSetupPos(0)
c 1757            |*:
c 1758               >
c 1759      ]
c 1760      ';'
c 1761      oMoveCmnts
c 1762      oSetupPos(0)
c 1763     }
  1764     oPopPosAndMark;                                          1484
  1765
  1766
  1767
  1768 CollectionTypeDefinition :                                  1219
```

```
1769
1770      oPushCopyPos(0)                                    1422
1771      oPushCopyMark(0)                                   1466
1772      ∂TypeDefinition                                    1250
1773      [=
1774          |'in':
1775             oPushCurrentTokenMark                       1555
1776             oSetupPos(0)                                1561
1777             'Ident'
1778             oPushCurrentTokenMark                       1555
1779             oSetupGcurPosNc                             1474
1780             oPushIndentPos(0)                           1537
1781             oPushNextTokenMark                          1547
1782             ∂Variable                                   1347
1783             oCcompress(0)                               1646
1784             oPopMark                                    1552
1785             oCcompress(0)                               1783
1786             oPopMark                                    1784
1787          |*:
1788      ]
1789      oPopPosAndMark;                                    1484
1790
1791
1792
1793 ProcedureDeclaration:                                   1061
1794
1795      'Ident'
1796      oPushIndentPos(0)                                  1780
1797      oSetupPos(0)                                       1561
1798      oCcompress(0)                                      1785
1799      oPushCopyPos(0)                                    1422
1800      oPushNextTokenMark                                 1781
1801      ∂FormalParameters                                     *
1802      oCcompress(0)                                      1798
1803      '='
1804      oPopPos                                            1558
1805      oSetupPos(0)                                       1797
1806      oPushCopyPos(0)                                    1799
1807      oPushNextTokenMark                                 1800
1808      ∂RoutineDefinition                                  313
1809      oPopPosAndMark;                                    1484
1810
1811
1812
1813 FunctionDeclaration :                                   1068
1814
1815      'Ident'
1816      oPushIndentPos(0)                                  1780
1817      oSetupPos(0)                                       1805
1818      oCcompress(0)                                      1802
1819      oPushCurrentTokenMark                              1778
1820      oPushCopyPos(0)                                    1806
1821      oPushNextTokenMark                                 1807
1822      ∂FormalParameters                                  1801
1823      [=
1824          |'returns':
```

```
1825              oPushCurrentTokenMark                                  1819
1826              oSetupPos(0)                                           1817
1827              'Ident'
1828              oSetupGcurPosNc                                        1779
1829              ':'
1830              oPushCurrentTokenMark                                  1819
1831              oSetupPos(0)                                           1817
1832              oPushIndentPos(0)                                      1780
1833              oPushNextTokenMark                                     1821
1834              ∂TypeDefinition                                        1772
1835              oCcompress(0)                                          1818
1836              oPopMark                                               1786
1837              oCcompress(0)                                          1835
1838              oPopMark                                               1836
1839          }*:
1840      ]
1841      '='
1842      oSetupGcurPosNc                                                1828
1843      oCcompress(0)                                                  1837
1844      oPopPosAndMark                                                 1809
1845      oCcompress(0)                                                  1837
1846      oPushCopyPos(0)                                                1820
1847      oPushNextTokenMark                                             1833
1848      ∂RoutineDefinition                                             1808
1849      oPopPosAndMark;                                                1844
1850
1851
1852
1853  FormalParameters :                                                 1822
1854
1855      [=
1856          |'(':
1857              oPushCurrentTokenMark                                  1830
1858              oPushCurrentTokenMark                                  1830
1859              oSetupPos(0)                                           1831
1860              oCcompress(0)                                          1845
1861              oPushCopyPos(0)                                        1846
1862              ∂RoutineFormal                                           *
1863              oCcompress(0)                                          1860
1864              {[=
c 1865                    |')':
c 1866                      oSetupPos(0)
c 1867                      oCcompress(0)
c 1868                      oPopMark
c 1869                      >
c 1870                    |',':
c 1871                       oPushCurrentTokenMark
c 1872                       oSetupPos(0)
c 1873                       oPushNextTokenMark
c 1874                       oPushCopyPos(0)
c 1875                       ∂RoutineFormal
c 1876                       oCcompress(0)
c 1877                       oPopMark
c 1878              ])
1879          }*:
1880      ]
```

```
1881      oPopPosAndMark;                                          1849
1882
1883
1884
1885  RoutineFormal :                                              1862
1886
1887      oPushCopyPos(0)                                          1861
1888      oPushCopyMark(0)                                         1771
1889      aPervasive                                               1015
1890      oCcompress(0)                                            1863
1891      oPushGcurPosNc                                           1546
1892      oPushNextTokenMark                                       1847
1893      aBindingCondition                                         509
1894      oCcompress(0)                                            1890
1895      oPushIndentPos(0)                                        1832
1896      oPushNextTokenMark                                       1892
1897      aIdentifierList                                          1329
1898      oCcompress(0)                                            1894
1899      ':'
1900      oPushCurrentTokenMark                                    1830
1901      oPushIndentPos(0)                                        1895
1902      oSetupPos(0)                                             1831
1903      oPushGcurPosNc                                           1891
1904      oPushNextTokenMark                                       1896
1905      aTypeDefinition                                          1834
1906      oCcompress(0)                                            1898
1907      oPopPosAndMark                                           1849
1908      oPopPosAndMark;                                          1907
1909
1910
1911
1912  RoutineDefinition :                                          1848
1913
1914      [[=
c 1915          |'imports':
c 1916             oPushCurrentTokenMark
c 1917             oPushIndentPos(0)
c 1918             oSetupPos(0)
c 1919             oCcompress(0)
c 1920             oPushIndentPos(0)
c 1921             oPushNextTokenMark
c 1922             aImportClause
c 1923             oCcompress(0)
c 1924             oPopPosAndMark
c 1925             ';'
c 1926             oMoveCmnts
c 1927             oSetupPos(0)
c 1928          |*:
c 1929             >
c 1930      ]]
1931        [=
1932          |'pre':
1933             oPushCurrentTokenMark                             1900
1934             oPushIndentPos(0)                                 1901
1935             oSetupPos(0)                                      1902
1936             [=
```

```
1937                |'(':                                                  
1938                   oPushCurrentTokenMark                        1900
1939                   oPushCurrentTokenMark                        1900
1940                   oPushIndentPos(0)                            1901
1941                   oSetupPos(0)                                 1935
1942                   oPushGcurPosNc                               1903
1943                   ∂Expression                                  1482
1944                   ')'
1945                   oSetupPos(0)                                 1941
1946                   oCcompress(0)                                1906
1947                   oPopPosAndMark                               1908
1948                   oCcompress(0)                                1946
1949                   oPopPosAndMark                               1947
1950                   ';'
1951                   oMoveCmnts                                   1560
1952                   oSetupPos(0)                                 1945
1953                |*:
1954                   oPopPosAndMark                               1949
1955                   ';'
1956                   oMoveCmnts                                   1951
1957                   oSetupPos(0)                                 1952
1958             ]
1959          |*:
1960       ]
1961       (=
1962          |'post':
1963             oPushCurrentTokenMark                             1900
1964             oPushIndentPos(0)                                 1901
1965             oSetupPos(0)                                      1952
1966             (=
1967                |'(':
1968                   oPushCurrentTokenMark                        1900
1969                   oPushCurrentTokenMark                        1900
1970                   oPushIndentPos(0)                            1901
1971                   oSetupPos(0)                                 1965
1972                   oPushGcurPosNc                               1942
1973                   ∂Expression                                  1943
1974                   ')'
1975                   oSetupPos(0)                                 1971
1976                   oCcompress(0)                                1948
1977                   oPopPosAndMark                               1949
1978                   oCcompress(0)                                1976
1979                   oPopPosAndMark                               1977
1980                   ';'
1981                   oMoveCmnts                                   1951
1982                   oSetupPos(0)                                 1975
1983                |*:
1984                   oPopPosAndMark                               1979
1985                   ';'
1986                   oMoveCmnts                                   1981
1987                   oSetupPos(0)                                 1982
1988             ]
1989          |*:
1990       ]
1991       oPushGcurPosNc                                          1972
1992       oPushNextTokenMark                                      1904
```

```
1993      @RoutineBody                                                    *
1994      oPopPosAndMark;                                                 1979
1995
1996
1997
1998  RoutineBody :                                                       1993
1999
2000      [=
2001          |'begin':
2002             oPushCurrentTokenMark                                    1900
2003             oSetupPos(0)                                             1982
2004             oCcompress(0)                                            1978
2005             oPushCopyPos(0)                                          1861
2006             oPushNextTokenMark                                       1904
2007             @BlockBody                                               *
2008             'end'
2009             oSetupPos(0)                                             1982
2010             [=
2011                 |'Ident':
2012                    oSetupGcurPosNc                                   1828
2013                 |*:
2014             ]
2015             oCcompress(0)                                            2004
2016             oPopMark                                                 1838
2017          |'code':
2018             oPushCurrentTokenMark                                    1900
2019             oSetupPos(0)                                             2009
2020             oCcompress(0)                                            2004
2021             [=
2022                 |'CodeStart':
2023                    oPushCurrentTokenMark                             1900
2024                    oPushIndentPos(0)                                 1901
2025                    oSetupPos(0)                                      2009
2026                    ([=
c 2027                        |'CodeMiddle':
c 2028                           oSetupPos(0)
c 2029                        |*:
c 2030                           'CodeEnd'
c 2031                           oSetupPos(0)
c 2032                           oCcompress(0)
c 2033                           oPopPosAndMark
c 2034                           >
c 2035                    )}
2036                 |*:
2037             ]
2038             'end'
2039             oSetupPos(0)                                             2025
2040             [=
2041                 |'Ident':
2042                    oSetupGcurPosNc                                   1828
2043                 |*:
2044             ]
2045             oCcompress(0)                                            2020
2046             oPopMark                                                 2016
2047          |'forward':
2048             oSetupPos(0)                                             2025
```

```
  2049              oCcompress(0)                                    2020
  2050           |'external':
  2051              oSetupPos(0)                                     2025
  2052              oCcompress(0)                                    2049
  2053       ]
  2054       oPopPosAndMark;                                         1979
  2055
  2056
  2057
  2058 Statements :                                                  *
  2059
  2060       [
c 2061        oPushIndentPos(0)
c 2062        [=
c 2063           |'Ident':
c 2064              oPushCurrentTokenMark
c 2065              oPushCurrentTokenMark
c 2066              oSetupPos(0)
c 2067              oCcompress(0)
c 2068              oPushCopyPos(0)
c 2069              ∂AssignmentOrCallStatement
c 2070              oCcompress(0)
c 2071              oPopMark
c 2072           |'exit':
c 2073              oPushCurrentTokenMark
c 2074              oPushCurrentTokenMark
c 2075              oSetupPos(0)
c 2076              oCcompress(0)
c 2077              oPushCopyPos(0)
c 2078              ∂ExitStatement
c 2079              oCcompress(0)
c 2080              oPopMark
c 2081           |'return':
c 2082              oPushCurrentTokenMark
c 2083              oPushCurrentTokenMark
c 2084              oSetupPos(0)
c 2085              oCcompress(0)
c 2086              oPushCopyPos(0)
c 2087              ∂ReturnStatement
c 2088              oCcompress(0)
c 2089              oPopMark
c 2090           |'assert':
c 2091              oPushCurrentTokenMark
c 2092              oSetupPos(0)
c 2093              oCcompress(0)
c 2094              oPushCopyPos(0)
c 2095              oPushNextTokenMark
c 2096              ∂AssertStatement
c 2097              oCcompress(0)
c 2098              oPopMark
c 2099           |'begin':
c 2100              oPushCurrentTokenMark
c 2101              oPushCurrentTokenMark
c 2102              oSetupPos(0)
c 2103              oCcompress(0)
c 2104              oPushCopyPos(0)
```

```
c 2105              @BeginStatement
c 2106              oCcompress(0)
c 2107              oPopMark
c 2108           |'if':
c 2109              oPushCurrentTokenMark
c 2110              oPushCurrentTokenMark
c 2111              oSetupPos(0)
c 2112              oCcompress(0)
c 2113              oPushCopyPos(0)
c 2114              @IfStatement
c 2115              oCcompress(0)
c 2116              oPopMark
c 2117           |'loop':
c 2118              oPushCurrentTokenMark
c 2119              oPushCurrentTokenMark
c 2120              oSetupPos(0)
c 2121              oCcompress(0)
c 2122              oPushCopyPos(0)
c 2123              @LoopStatement
c 2124              oCcompress(0)
c 2125              oPopMark
c 2126           |'for':
c 2127              oPushCurrentTokenMark
c 2128              oPushCurrentTokenMark
c 2129              oSetupPos(0)
c 2130              oCcompress(0)
c 2131              oPushCopyPos(0)
c 2132              @ForLoopStatement
c 2133              oCcompress(0)
c 2134              oPopMark
c 2135           |'case':
c 2136              oPushCurrentTokenMark
c 2137              oPushCurrentTokenMark
c 2138              oSetupPos(0)
c 2139              oCcompress(0)
c 2140              oPushCopyPos(0)
c 2141              @CaseStatement
c 2142              oCcompress(0)
c 2143              oPopMark
c 2144           |';':
c 2145              oMoveCmnts
c 2146              oSetupPos(0)
c 2147           |*:
c 2148              oPopPos
c 2149              >
c 2150        ]
c 2151        oCcompress(0)
c 2152        oPopPosAndMark
c 2153        ';'
c 2154        oMoveCmnts
c 2155        oPushCurrentTokenMark
c 2156        oSetupPos(0)
c 2157       }
  2158       oPopPosAndMark;
  2159
  2160
```

```
2161
2162 AssignmentOrCallStatement :                                    *
2163
2164     oPushIndentPos(0)                                          1901
2165     oPushCopyMark(0)                                           1888
2166     ∂Variable                                                  1782
2167     oCcompress(0)                                              2052
2168     [=
2169         |':=':
2170             oPushCurrentTokenMark                              1900
2171             oPushCurrentTokenMark                              1900
2172             oSetupPos(0)                                       2025
2173             oPushGcurPosNc                                     1972
2174             ∂Expression                                        1973
2175             oCcompress(0)                                      2167
2176             oPopMark                                           2046
2177         |*:
2178     ]
2179     oPopPosAndMark;                                            2054
2180
2181
2182
2183 ExitStatement :                                                *
2184
2185     [=
2186         |'when':
2187             oPushGcurPosNc                                     2173
2188             oSetupPos(0)                                       2025
2189             oCcompress(0)                                      2175
2190             oPushGcurPosNc                                     2173
2191             oPushNextTokenMark                                 2006
2192             ∂Expression                                        2174
2193             oPopPos                                            1804
2194         |*:
2195     ]
2196     oPopPosAndMark;                                            2054
2197
2198
2199
2200 ReturnStatement :                                              *
2201
2202     [=
2203         |'(':
2204             oPushCurrentTokenMark                              1900
2205             oPushCurrentTokenMark                              1900
2206             oPushIndentPos(0)                                  1901
2207             oSetupPos(0)                                       2025
2208             oCcompress(0)                                      2189
2209             oPushGcurPosNc                                     2190
2210             ∂Expression                                        2192
2211             ')'
2212             oSetupPos(0)                                       2207
2213             oCcompress(0)                                      2208
2214             oPopPosAndMark                                     2054
2215             oCcompress(0)                                      2213
2216         |*:
```

```
2217      ]
2218      [=
2219          |'when':
2220             oPushIndentPos(0)                          1901
2221             oSetupPos(0)                               2212
2222             oPushNextTokenMark                         2191
2223             oPushGcurPosNc                             2209
2224             @Expression                                2210
2225             oPopPos                                    2193
2226          |*:
2227      ]
2228      oPopPosAndMark;                                   2214
2229
2230
2231
2232  AssertStatement :                                      *
2233
2234      [=
2235          |'(':
2236             oPushCurrentTokenMark                      1900
2237             oPushCurrentTokenMark                      1900
2238             oPushIndentPos(0)                          1901
2239             oSetupPos(0)                               2221
2240             oPushGcurPosNc                             2223
2241             @Expression                                2224
2242             ')'
2243             oSetupPos(0)                               2239
2244             oCcompress(0)                              2215
2245             oPopPosAndMark                             2214
2246          |*:
2247      ]
2248      oPopPosAndMark;                                   2245
2249
2250
2251
2252  BeginStatement :                                       *
2253
2254      oPushCopyPos(0)                                   2005
2255      oPushCopyMark(0)                                  2165
2256      @BlockBody                                        2007
2257      oCcompress(0)                                     2244
2258      'end'
2259      oSetupPos(0)                                      2243
2260      oPopPosAndMark;                                   2245
2261
2262
2263
2264
2265
2266
2267  BlockBody :                                           2256
2268
2269      [=
2270          |'not':
2271             oPushIndentPos(0)                          1901
2272             oSetupPos(0)                               2259
```

```
2273            oCcompress(0)                                   2257
2274            'checked'
2275            oSetupGcurPosNc                                 1828
2276            ';'
2277            oPopPos                                         2225
2278            oMoveCmnts                                      1981
2279            oSetupPos(0)                                    2272
2280        |'checked':
2281            oPushIndentPos(0)                               1901
2282            oSetupPos(0)                                    2279
2283            oCcompress(0)                                   2273
2284            ';'
2285            oPopPos                                         2277
2286            oMoveCmnts                                      2278
2287            oSetupPos(0)                                    2282
2288        |*:
2289    ]
2290    oPushCopyPos(0)                                         2005
2291    oPushCopyMark(0)                                        2255
2292    ∂Declarations                                            546
2293    oPushCopyPos(0)                                         2005
2294    oPushCopyMark(0)                                        2291
2295    ∂Statements                                             2058
2296    oPopPosAndMark;                                         2260
2297
2298
2299
2300 IfStatement :                                               *
2301
2302    oPushGcurPosNc                                          2240
2303    oPushCopyMark(0)                                        2294
2304    ∂Expression                                             2241
2305    oCcompress(0)                                           2283
2306    'then'
2307    oPushCurrentTokenMark                                   1900
2308    oSetupPos(0)                                            2287
2309    oPushCopyPos(0)                                         2293
2310    ∂BlockBody                                              2267
2311    oCcompress(0)                                           2305
2312    oPushCopyPos(0)                                         2309
2313    oPushNextTokenMark                                      2222
2314    ∂ElseClause                                              *
2315    oCcompress(0)                                           2311
2316    'end'
2317    oSetupPos(0)                                            2308
2318    'if'
2319    oSetupGcurPosNc                                         2275
2320    oPopPosAndMark;                                         2296
2321
2322
2323
2324 ElseClause :                                               2314
2325
2326    [=
2327        |'elseif':
2328            oPushCurrentTokenMark                           2307
```

```
2329            oSetupPos(0)                                    2317
2330            oCcompress(0)                                   2315
2331            oPushGcurPosNc                                  2240
2332            oPushNextTokenMark                              2313
2333            @Expression                                     2304
2334            oCcompress(0)                                   2330
2335            'then'
2336            oPushCurrentTokenMark                           2307
2337            oPushIndentPos(0)                               1901
2338            oSetupPos(0)                                    2317
2339            oPushGcurPosNc                                  2331
2340            oPushNextTokenMark                              2332
2341            @BlockBody                                      2310
2342            oCcompress(0)                                   2334
2343            oPopMark                                        2176
2344            oCcompress(0)                                   2342
2345            oPushCopyPos(0)                                 2312
2346            oPushNextTokenMark                              2340
2347            @ElseClause                                     2324
2348            oCcompress(0)                                   2344
2349            oPopPosAndMark                                  2320
2350
2351        |'else':
2352            oPushCurrentTokenMark                           2336
2353            oPushCurrentTokenMark                           2336
2354            oSetupPos(0)                                    2338
2355            oCcompress(0)                                   2348
2356            oPushCopyPos(0)                                 2345
2357            @BlockBody                                      2341
2358            oCcompress(0)                                   2355
2359            oPopMark                                        2343
2360        |*:
2361    ]
2362    oPopPosAndMark;                                         2349
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373 LoopStatement :                                             *
2374
2375    oPushCopyPos(0)                                         2356
2376    oPushCopyMark(0)                                        2294
2377    @BlockBody                                              2357
2378    'end'
2379    oSetupPos(0)                                            2338
2380    'loop'
2381    oSetupGcurPosNc                                         2319
2382    oPopPosAndMark;                                         2349
2383
2384
```

```
2385
2386 ForLoopStatement :                                                    *
2387
2388     'Ident'
2389     oPushCurrentTokenMark                                          2336
2390     oPushIndentPos(0)                                              2337
2391     oSetupPos(0)                                                   2379
2392     oCcompress(0)                                                  2358
2393     [=
2394         |'decreasing':
2395            oSetupGcurPosNc                                         2381
2396         |*:
2397     ]
2398     'in'
2399     oPushCurrentTokenMark                                          2336
2400     oPushCurrentTokenMark                                          2336
2401     oSetupGcurPosNc                                                2381
2402     oPushGcurPosNc                                                 2339
2403     oPushGcurPosNc                                                 2339
2404     @Expression                                                    2333
2405     [=
2406         |'..':
2407            oPushCurrentTokenMark                                   2336
2408            oPushCurrentTokenMark                                   2336
2409            oSetupPos(0)                                            2391
2410            oPushGcurPosNc                                          2339
2411            @Sum                                                    1453
2412            oCcompress(0)                                           2392
2413            oPopMark                                                2359
2414            oCcompress(0)                                           2412
2415         |*:
2416     ]
2417     oPopMark                                                       2413
2418     oCcompress(0)                                                  2414
2419     oPopPosAndMark                                                 2382
2420     oCcompress(0)                                                  2414
2421     'loop'
2422     oSetupPos(0)                                                   2391
2423     oPushCopyPos(0)                                                2356
2424     oPushNextTokenMark                                             2346
2425     @LoopStatement                                                 2373
2426     oPopPos                                                        2285
2427     oPopPosAndMark;                                                2419
2428
2429
2430
2431 CaseStatement :                                                     *
2432
2433     [=
2434         |'var':
2435            oPushIndentPos(0)                                       2337
2436            oSetupPos(0)                                            2422
2437            oCcompress(0)                                           2420
2438            'Ident'
2439            oSetupGcurPosNc                                         2381
2440            oCcompress(0)                                           2437
```

```
2441            'bound'
2442            oPushCurrentTokenMark                   2336
2443            oSetupPos(0)                            2436
2444            'to'
2445            oSetupGcurPosNc                         2439
2446            'Ident'
2447            oPushCurrentTokenMark                   2442
2448            oSetupGcurPosNc                         2445
2449            oPushGcurPosNc                          2410
2450            oPushNextTokenMark                      2424
2451            aVariable                               2166
2452            oCcompress(0)                           2440
2453            oPopMark                                2413
2454            oCcompress(0)                           2452
2455            oPopMark                                2453
2456            oCcompress(0)                           2454
2457            oPopPos                                 2426
2458          |'readonly':
2459            oPushIndentPos(0)                       2337
2460            oSetupPos(0)                            2443
2461            oCcompress(0)                           2456
2462            'Ident'
2463            oSetupGcurPosNc                         2448
2464            oCcompress(0)                           2461
2465            'bound'
2466            oPushCurrentTokenMark                   2447
2467            oSetupPos(0)                            2460
2468            'to'
2469            oSetupGcurPosNc                         2463
2470            'Ident'
2471            oPushCurrentTokenMark                   2466
2472            oSetupGcurPosNc                         2469
2473            oPushGcurPosNc                          2449
2474            oPushNextTokenMark                      2450
2475            aVariable                               2451
2476            oCcompress(0)                           2464
2477            oPopMark                                2455
2478            oCcompress(0)                           2476
2479            oPopMark                                2477
2480            oCcompress(0)                           2478
2481            oPopPos                                 2457
2482          |'const':
2483            oPushIndentPos(0)                       2337
2484            oSetupPos(0)                            2467
2485            oCcompress(0)                           2480
2486            'Ident'
2487            oSetupGcurPosNc                         2472
2488            oCcompress(0)                           2485
2489            ':='
2490            oPushCurrentTokenMark                   2471
2491            oSetupPos(0)                            2484
2492            'Ident'
2493            oSetupGcurPosNc                         2487
2494            oPushGcurPosNc                          2473
2495            oPushNextTokenMark                      2474
2496            aVariable                               2475
```

```
2497            oCcompress(0)                                   2488
2498            oPopMark                                        2479
2499            oCcompress(0)                                   2497
2500            oPopPos                                         2481
2501         |'Ident':
2502            oPushIndentPos(0)                               2337
2503            oSetupPos(0)                                    2491
2504            oCcompress(0)                                   2499
2505            [=
2506              |':=':
2507                   oPushCurrentTokenMark                    2490
2508                   oSetupPos(0)                             2503
2509                   'Ident'
2510                   oSetupGcurPosNc                          2493
2511                   oPushGcurPosNc                           2494
2512                   oPushNextTokenMark                       2495
2513                   ∂Variable                                2496
2514                   oCcompress(0)                            2504
2515                   oPopMark                                 2498
2516               |'bound':
2517                   oPushCurrentTokenMark                    2490
2518                   oSetupPos(0)                             2503
2519                   'to'
2520                   oSetupGcurPosNc                          2510
2521                   'Ident'
2522                   oPushCurrentTokenMark                    2490
2523                   oSetupGcurPosNc                          2520
2524                   oPushGcurPosNc                           2511
2525                   oPushNextTokenMark                       2512
2526                   ∂Variable                                2513
2527                   oCcompress(0)                            2514
2528                   oPopMark                                 2515
2529                   oCcompress(0)                            2527
2530                   oPopMark                                 2528
2531               |*:
2532                   oPushGcurPosNc                           2524
2533                   oPushNextTokenMark                       2525
2534                   ∂Variable                                2526
2535                   oCcompress(0)                            2529
2536                   oPushCopyPos(0)                          2423
2537                   oPushNextTokenMark                       2525
2538                   ∂PartialSum                              1424
2539            ]
2540            oCcompress(0)                                   2529
2541            oPopPos                                         2500
2542         |*:
2543            oPushIndentPos(0)                               2337
2544            oPushCopyMark(0)                                2376
2545            ∂Sum                                            2411
2546            oCcompress(0)                                   2540
2547      ]
2548      'of'
2549      oSetupPos(0)                                          2503
2550      oPushCopyPos(0)                                       2536
2551      oPushNextTokenMark                                    2537
2552      ∂CaseBody                                               *
```

```
  2553      oPopPosAndMark;                                                2427
  2554
  2555
  2556
  2557 CaseBody :                                                          2552
  2558
  2559      ([=
c 2560           |'end':
c 2561              oSetupPos(0)
c 2562              'case'
c 2563              oSetupGcurPosNc
c 2564              >
c 2565           |'otherwise':
c 2566              oPushCurrentTokenMark
c 2567              oPushGcurPosNc
c 2568              oSetupPos(0)
c 2569              '=>'
c 2570              oPushCurrentTokenMark
c 2571              oSetupPos(0)
c 2572              oPushGcurPosNc
c 2573              oPushNextTokenMark
c 2574              @BlockBody
c 2575              oCcompress(0)
c 2576              oPopPosAndMark
c 2577              oCcompress(0)
c 2578              oPopMark
c 2579              'end'
c 2580              oSetupPos(0)
c 2581              'case'
c 2582              oSetupGcurPosNc
c 2583              >
c 2584           |*:
c 2585              oPushCurrentTokenMark
c 2586              oPushGcurPosNc
c 2587              oPushGcurPosNc
c 2588              oPushCurrentTokenMark
c 2589              @CaseLabel
c 2590              ([=
c 2591                   |',':
c 2592                      oPushCurrentTokenMark
c 2593                      oSetupPos(0)
c 2594                      oPushGcurPosNc
c 2595                      oPushNextTokenMark
c 2596                      @CaseLabel
c 2597                      oCcompress(0)
c 2598                      oPopMark
c 2599                      oCcompress(0)
c 2600                   |*:
c 2601                      >
c 2602              ])
  2603              oCcompress(0)                                          2546
  2604              '=>'
  2605              oPushCurrentTokenMark                                  2522
  2606              oSetupPos(0)                                           2549
  2607              oPushGcurPosNc                                         2524
  2608              oPushNextTokenMark                                     2551
```

```
2609                ∂BlockBody                                  2377
2610                oCcompress(0)                               2546
2611                oPopMark                                    2530
2612                oCcompress(0)                               2610
2613                'end'
2614                oPushCurrentTokenMark                       2605
2615                oSetupPos(0)                                2606
2616                oPushGcurPosNc                              2607
2617                oPushNextTokenMark                          2608
2618                ∂CaseLabel                                  1548
2619                oCcompress(0)                               2612
2620                oPopMark                                    2611
2621                oCcompress(0)                               2619
2622                ';'
2623                oPopPosAndMark                              2553
2624                oMoveCmnts                                  2286
2625                oSetupPos(0)                                2615
2626        ]}
2627        oPopPosAndMark;                                     2623
2628
2629
2630
2631 CaseLabel :                                                2618
2632
2633        oPushCopyPos(0)                                     2550
2634        oPushCopyMark(0)                                    2544
2635        ∂Expression                                         2333
2636        oCcompress(0)                                       2621
2637        [=
2638            |'..':
2639                oPushCurrentTokenMark                       2614
2640                oSetupPos(0)                                2625
2641                oPushGcurPosNc                              2616
2642                oPushNextTokenMark                          2617
2643                ∂Sum                                        2545
2644                oCcompress(0)                               2636
2645                oPopMark                                    2620
2646            |*:
2647        ]
2648        oCcompress(0)                                       2644
2649        oPopPosAndMark;                                     2627
2650
2651
2652
2653 Expression :                                               2635
2654
2655        oPushCopyPos(0)                                     2550
2656        oPushCopyMark(0)                                    2634
2657        ∂SubExpression                                        *
2658        oPopPosAndMark;                                     2649
2659
2660
2661
2662 SubExpression :                                            2657
2663
2664        oPushCopyPos(0)                                     2550
```

```
  2665      oPushCopyMark(0)                                          2656
  2666      @Disjunction                                                *
  2667      [=
  2668          |'->':
  2669             oPushCurrentTokenMark                              2614
  2670             oSetupPos(0)                                       2625
  2671             oPushGcurPosNc                                     2641
  2672             oPushNextTokenMark                                 2642
  2673             @Disjunction                                       2666
  2674             oCcompress(0)                                      2644
  2675             oPopMark                                           2645
  2676          |*:
  2677      ]
  2678      oCcompress(0)                                             2674
  2679      oPopPosAndMark;                                           2658
  2680
  2681
  2682
  2683 Disjunction :                                                  2673
  2684
  2685      oPushCopyPos(0)                                           2550
  2686      oPushCopyMark(0)                                          2665
  2687      @Conjunction                                                *
  2688      {[=
c 2689           |'or':
c 2690              oPushCurrentTokenMark
c 2691              oSetupPos(0)
c 2692              oPushGcurPosNc
c 2693              oPushNextTokenMark
c 2694              @Conjunction
c 2695              oCcompress(0)
c 2696              oPopMark
c 2697           |*:
c 2698              oCcompress(0)
c 2699              >
c 2700      ]}
  2701      oPopPosAndMark;                                           2679
  2702
  2703
  2704
  2705 Conjunction :                                                  2687
  2706
  2707      oPushCopyPos(0)                                           2550
  2708      oPushCopyMark(0)                                          2686
  2709      @Negation                                                   *
  2710      {[=
c 2711           |'and':
c 2712              oPushCurrentTokenMark
c 2713              oSetupPos(0)
c 2714              oPushGcurPosNc
c 2715              oPushNextTokenMark
c 2716              @Negation
c 2717              oCcompress(0)
c 2718              oPopMark
c 2719           |*:
c 2720              oCcompress(0)
```

```
c 2721              >
c 2722      ])
  2723      oPopPosAndMark;                                   2701
  2724
  2725
  2726
  2727  Negation :                                            2709
  2728
  2729      [=
  2730          |'not':
  2731              oPushCurrentTokenMark                     2614
  2732              oSetupPos(0)                              2625
  2733              oCcompress(0)                             2674
  2734              oPushGcurPosNc                            2671
  2735              oPushNextTokenMark                        2672
  2736              ∂Relation                                   *
  2737              oCcompress(0)                             2733
  2738              oPopMark                                  2675
  2739          |*:
  2740              oPushCopyPos(0)                           2550
  2741              oPushCopyMark(0)                          2708
  2742              ∂Relation                                 2736
  2743      ]
  2744      oPopPosAndMark;                                   2723
  2745
  2746
  2747
  2748  Relation :                                            2742
  2749
  2750      oPushCopyPos(0)                                   2550
  2751      oPushCopyMark(0)                                  2741
  2752      ∂SubSum                                             *
  2753      oCcompress(0)                                     2737
  2754      [=
  2755          |'=':
  2756              oPushCurrentTokenMark                     2614
  2757              oPushCurrentTokenMark                     2614
  2758              oSetupPos(0)                              2625
  2759              oPushGcurPosNc                            2734
  2760              ∂SubSum                                   2752
  2761              oCcompress(0)                             2753
  2762              oPopMark                                  2738
  2763          |'<':
  2764              oPushCurrentTokenMark                     2614
  2765              oPushCurrentTokenMark                     2614
  2766              oSetupPos(0)                              2625
  2767              oPushGcurPosNc                            2759
  2768              ∂SubSum                                   2760
  2769              oCcompress(0)                             2761
  2770              oPopMark                                  2762
  2771          |'<=':
  2772              oPushCurrentTokenMark                     2614
  2773              oPushCurrentTokenMark                     2614
  2774              oSetupPos(0)                              2625
  2775              oPushGcurPosNc                            2767
  2776              ∂SubSum                                   2768
```

```
2777            oCcompress(0)                                   2769
2778            oPopMark                                        2770
2779        |'>':
2780            oPushCurrentTokenMark                           2614
2781            oPushCurrentTokenMark                           2614
2782            oSetupPos(0)                                    2625
2783            oPushGcurPosNc                                  2775
2784            aSubSum                                         2776
2785            oCcompress(0)                                   2777
2786            oPopMark                                        2778
2787        |'>=':
2788            oPushCurrentTokenMark                           2614
2789            oPushCurrentTokenMark                           2614
2790            oSetupPos(0)                                    2625
2791            oPushGcurPosNc                                  2783
2792            aSubSum                                         2784
2793            oCcompress(0)                                   2785
2794            oPopMark                                        2786
2795        |'not':
2796            oPushCurrentTokenMark                           2614
2797            oSetupPos(0)                                    2625
2798            [=
2799                |'=':
2800                    oPushCurrentTokenMark                   2614
2801                    oSetupGcurPosNc                         2523
2802                    oPushGcurPosNc                          2791
2803                    aSubSum                                 2792
2804                |'in':
2805                    oPushCurrentTokenMark                   2614
2806                    oPushCurrentTokenMark                   2614
2807                    oSetupGcurPosNc                         2523
2808                    oPushGcurPosNc                          2791
2809                    oPushGcurPosNc                          2791
2810                    aSum                                    2643
2811                    [=
2812                        |'..':
2813                            oPushCurrentTokenMark           2614
2814                            oPushCurrentTokenMark           2614
2815                            oSetupPos(0)                    2625
2816                            oPushGcurPosNc                  2791
2817                            aSum                            2643
2818                            oCcompress(0)                   2793
2819                            oPopMark                        2794
2820                            oCcompress(0)                   2818
2821                        |*:
2822
2823
2824                    ]
2825                    oPopPosAndMark                          2744
2826            ]
2827            oCcompress(0)                                   2820
2828            oPopMark                                        2819
2829        |'in':
2830            oPushCurrentTokenMark                           2614
2831            oPushCurrentTokenMark                           2614
2832            oSetupPos(0)                                    2625
```

```
2833            oPushGcurPosNc                                  2816
2834            oPushGcurPosNc                                  2833
2835            ∂Sum                                            2817
2836            [=
2837                {'..':
2838                    oPushCurrentTokenMark                   2614
2839                    oPushCurrentTokenMark                   2614
2840                    oSetupPos(0)                            2625
2841                    oPushGcurPosNc                          2834
2842                    ∂Sum                                    2835
2843                    oCcompress(0)                           2820
2844                    oPopMark                                2828
2845                #:
2846            ]
2847            oCcompress(0)                                   2843
2848            oPopPosAndMark                                  2744
2849        #:
2850    ]
2851    oCcompress(0)                                           2843
2852    oPopPosAndMark;                                         2848
2853
2854
2855
2856 Sum :                                                      2842
2857
2858    oPushCopyPos(0)                                         2550
2859    oPushCopyMark(0)                                        2751
2860    ∂SubSum                                                 2792
2861    oCcompress(0)                                           2843
2862    oPopPosAndMark;                                         2852
2863
2864
2865
2866 SubSum :                                                   2860
2867
2868    oPushCopyPos(0)                                         2550
2869    oPushCopyMark(0)                                        2859
2870    ∂Factor                                                 1420
2871    oCcompress(0)                                           2861
2872    oPushCopyPos(0)                                         2550
2873    oPushNextTokenMark                                      2735
2874    ∂PartialSubSum                                            *
2875    oPopPosAndMark;                                         2862
2876
2877
2878
2879 PartialSum :                                               2538
2880
2881    oPushCopyPos(0)                                         2872
2882    oPushCopyMark(0)                                        2869
2883    ∂PartialSubSum                                          2874
2884    oCcompress(0)                                           2871
2885    oPopPosAndMark;                                         2875
2886
2887
2888
```

```
  2889 PartialSubSum :                                                  2883
  2890
  2891     oPushCopyPos(0)                                              2872
  2892     oPushCopyMark(0)                                             2882
  2893     @PartialTerm                                                   *
  2894     ((=
c 2895          |'+':
c 2896             oPushCurrentTokenMark
c 2897             oPushCurrentTokenMark
c 2898             oSetupPos(0)
c 2899             oPushGcurPosNc
c 2900             @Term
c 2901             oCcompress(0)
c 2902             oPopMark
c 2903          |'-':
c 2904             oPushCurrentTokenMark
c 2905             oPushCurrentTokenMark
c 2906             oSetupPos(0)
c 2907             oPushGcurPosNc
c 2908             @Term
c 2909             oCcompress(0)
c 2910             oPopMark
c 2911          |'xor':
c 2912             oPushCurrentTokenMark
c 2913             oPushCurrentTokenMark
c 2914             oSetupPos(0)
c 2915             oPushGcurPosNc
c 2916             @Term
c 2917             oCcompress(0)
c 2918             oPopMark
c 2919          |*:
c 2920             oCcompress(0)
c 2921             >
c 2922     ))
  2923     oPopPosAndMark;                                              2885
  2924
  2925
  2926
  2927 Term :                                                           1384
  2928
  2929     oPushCopyPos(0)                                              2872
  2930     oPushCopyMark(0)                                             2892
  2931     @Factor                                                      2870
  2932     oCcompress(0)                                                2884
  2933     oPushCopyPos(0)                                              2872
  2934     oPushNextTokenMark                                           2873
  2935     @PartialTerm                                                 2893
  2936     oPopPosAndMark;                                              2923
  2937
  2938
  2939
  2940 PartialTerm :                                                   2935
  2941
  2942     ((=
c 2943          |'*':
c 2944             oPushCurrentTokenMark
```

```
c 2945              oPushCurrentTokenMark
c 2946              oSetupPos(0)
c 2947              oPushGcurPosNc
c 2948              @Factor
c 2949              oCcompress(0)
c 2950              oPopMark
c 2951           |'div':
c 2952              oPushCurrentTokenMark
c 2953              oPushCurrentTokenMark
c 2954              oSetupPos(0)
c 2955              oPushGcurPosNc
c 2956              @Factor
c 2957              oCcompress(0)
c 2958              oPopMark
c 2959           |'mod':
c 2960              oPushCurrentTokenMark
c 2961              oPushCurrentTokenMark
c 2962              oSetupPos(0)
c 2963              oPushGcurPosNc
c 2964              @Factor
c 2965              oCcompress(0)
c 2966              oPopMark
c 2967           |*:
c 2968              oCcompress(0)
c 2969              >
c 2970     ])
  2971     oPopPosAndMark;                                          2936
  2972
  2973
  2974
  2975 Factor :                                                    2931
  2976
  2977     (=
  2978         |'Ident':
  2979            oSetupPos(0)                                     2625
  2980            oCcompress(0)                                    2932
  2981            oPushIndentPos(0)                                2337
  2982            oPushCopyMark(0)                                 2930
  2983            @Variable                                        2526
  2984            oCcompress(0)                                    2980
  2985         |'Number','OctalNumber','HexNumber':
  2986            oSetupPos(0)                                     2625
  2987            oCcompress(0)                                    2984
  2988         |'Char','ExChar':
  2989            oSetupPos(0)                                     2625
  2990            oCcompress(0)                                    2987
  2991         |'MDChar':
  2992            oSetupPos(0)                                     2625
  2993            oCcompress(0)                                    2990
  2994         |'String','ExString':
  2995            oSetupPos(0)                                     2625
  2996            oCcompress(0)                                    2993
  2997         |'MDString':
  2998            oSetupPos(0)                                     2625
  2999            oCcompress(0)                                    2996
  3000         |'(':
```

```
  3001              oPushCurrentTokenMark                          2614
  3002              oPushCurrentTokenMark                          2614
  3003              oPushGcurPosNc                                 2841
  3004              oSetupPos(0)                                   2625
  3005              oCcompress(0)                                  2999
  3006              oPushGcurPosNc                                 2841
  3007              @SubExpression                                 2662
  3008              ')'
  3009              oSetupPos(0)                                   2625
  3010              oCcompress(0)                                  3005
  3011              oPopPosAndMark                                 2936
  3012           |'-':
  3013              oPushCurrentTokenMark                          2614
  3014              oSetupPos(0)                                   3009
  3015              oCcompress(0)                                  3010
  3016              oPushNextTokenMark                             2934
  3017              oPushGcurPosNc                                 3006
  3018              @Factor                                        2975
  3019              oCcompress(0)                                  3015
  3020              oPopMark                                       2844
  3021        ]
  3022        oPopPosAndMark;                                      3011
  3023
  3024
  3025
  3026  Variable :                                                 2983
  3027
  3028        {[=
c 3029           |'(':
c 3030              oPushCurrentTokenMark
c 3031              oPushCurrentTokenMark
c 3032              oSetupPos(0)
c 3033              oCcompress(0)
c 3034              oPushCopyPos(0)
c 3035              @Subscripts
c 3036              oCcompress(0)
c 3037              oPopMark
c 3038           |'.':
c 3039              oSetupPos(0)
c 3040              'Ident'
c 3041              oSetupGcurPosNc
c 3042              oCcompress(0)
c 3043           |'^':
c 3044              oSetupPos(0)
c 3045              oCcompress(0)
c 3046           |*:
c 3047              oCcompress(0)
c 3048              >
c 3049        ]}
  3050        oPopPosAndMark;                                      3022
  3051
  3052
  3053
  3054  Subscripts :                                                *
  3055
  3056        {
```

```
c 3057        [=
c 3058            ['(']:
c 3059              oSetupPos(0)
c 3060              >
c 3061            ['parameter']:
c 3062               oPushGcurPosNc
c 3063               oSetupPos(0)
c 3064               oPopPos
c 3065            ['any']:
c 3066               oPushGcurPosNc
c 3067               oSetupPos(0)
c 3068               oPopPos
c 3069            ['unknown']:
c 3070               oPushGcurPosNc
c 3071               oSetupPos(0)
c 3072               oPopPos
c 3073            ['all']:
c 3074               oPushGcurPosNc
c 3075               oSetupPos(0)
c 3076               oPopPos
c 3077            [*:
c 3078               oPushGcurPosNc
c 3079               oPushCopyMark(0)
c 3080               ∂Expression
c 3081               oCcompress(0)
c 3082               [=
c 3083                   ['..']:
c 3084                      oPushCurrentTokenMark
c 3085                      oSetupPos(0)
c 3086                      oPushNextTokenMark
c 3087                      oPushGcurPosNc
c 3088                      ∂Sum
c 3089                      oCcompress(0)
c 3090                      oPopMark
c 3091                   ]*:
c 3092               ]
c 3093        ]
c 3094        [=
c 3095            [')']:
c 3096               oSetupPos(0)
c 3097               >
c 3098            [',']:
c 3099               oSetupPos(0)
c 3100        ]
c 3101     }
  3102     oPopPosAndMark;                                          3022
  3103
  3104
  3105 _|_
  3106
  3107
  3108
  3109
```

cross reference * is def = is asg

-A-
AssertStatement 2232*
AssignmentOrCallStatem- 2162*
ent
Author 7*

-B-
BeginStatement 2252*
BindingCondition 384* 405* 509* 1893*
BlockBody 2007* 2256* 2267* 2310* 2341* 2357* 2377* 2609*

-C-
CaseBody 2552* 2557*
CaseLabel 1548* 2618* 2631*
CaseStatement 2431*
CollectionTypeDefiniti- 1178* 1190* 1203* 1219* 1768*
on
Conjunction 2687* 2705*
ConstantDeclaration 819*
ConverterDeclaration 897*
Cordy 7
Crowe 8

-D-
D 8
Date 9
Declarations 233* 546* 2292*
DeclarationsInRecord 1467* 1541* 1568*
Disjunction 2666* 2673* 2683*

-E-
ElseClause 2314* 2324* 2347*
EnumeratedTypeDefiniti- 1085* 1312* 1325*
on
Euclid 4* 6*
ExitStatement 2183*
ExportClause 204* 449* 536*
ExportItem 457* 474*
Expression 274* 294* 711 738* 831* 856 961 1146* 1194* 1482* 1943* 1973* 2174* 2192* 2210* 2224*
2241* 2304* 2333* 2404 2635* 2653*

-F-
Factor 1396* 1408* 1420* 2870* 2931* 2975* 3018*
Feb 9
ForLoopStatement 2386*
FormalParameters 1801* 1822* 1853*
FunctionDeclaration 258* 1068* 1813*

-I-
IdentifierList 719* 823* 991* 1024* 1329* 1897*
IfStatement 2300*
ImportClause 327*
ImportItem 335* 376*

IndexType 1032* 1092* 1126* 1243* 1276* 1304*
InlineProcedureDeclara- 1052*
tion
Inputs 13

-J-
J 7

-L-
LoopStatement 2373* 2425*

-M-
MDRecordBody 1154* 1657*
ModuleBody 173* 1114* 1164* 1265* 1287*
Modules 171*
ModuleTrailer 1117* 1167* 1268* 1290* 1642*

-N-
NameOrRangeTypeDefinit- 1296* 1317* 1338*
ion
Negation 2709* 2727*

-O-
oCcompress 144 162* 177* 205* 242* 259* 277* 279* 297* 299* 314* 321 332* 387* 398* 417
428* 443* 454* 480* 483* 486* 490* 503* 516* 532* 537* 701* 722 724* 731* 739*
741* 791* 799* 814 824* 837* 851* 858 861* 902* 917* 923* 933* 944* 964* 968
985* 996* 1005* 1019 1025* 1033* 1035* 1058* 1065* 1070* 1084* 1089* 1102* 1112* 1121* 1131*
1147* 1173* 1182* 1195* 1204* 1210* 1220* 1225* 1232* 1237* 1244* 1251* 1277* 1298* 1319* 1330*
1333* 1344* 1348* 1361* 1365* 1373* 1377* 1385* 1389* 1397* 1401* 1409* 1413* 1421* 1425* 1439
1441* 1454* 1456* 1483* 1485* 1542* 1549* 1551* 1631* 1646* 1783* 1785* 1798* 1802* 1818* 1835*
1837* 1843 1845* 1860* 1863* 1890* 1894* 1898* 1906* 1946* 1948* 1976* 1978* 2004* 2015 2020*
2045 2049* 2052* 2167* 2175* 2189* 2208* 2213* 2215* 2244* 2257* 2273* 2283* 2305* 2311* 2315*
2330* 2334* 2342* 2344* 2348* 2355* 2358* 2392* 2412* 2414* 2418 2420* 2437* 2440* 2452* 2454*
2456* 2461* 2464* 2476* 2478* 2480* 2485* 2488* 2497* 2499* 2504* 2514* 2527* 2529* 2535 2540*
2546* 2603 2610* 2612* 2619* 2621* 2636* 2644* 2648 2674* 2678 2733* 2737* 2753* 2761* 2769*
2777* 2785* 2793* 2818* 2820* 2827 2843* 2847 2851 2861* 2871* 2884* 2932* 2980* 2984* 2987*
2990* 2993* 2996* 2999* 3005* 3010* 3015* 3019*
oMoveCmnts 136 165* 208* 220* 227* 245* 262* 282* 302* 317* 971* 1554* 1560* 1951* 1956 1981*
1986 2278* 2286* 2624*
oPopMark 135 163* 322* 388* 418* 504* 538* 541 723* 740* 744 859* 862* 924* 969* 986*
1006* 1071* 1205* 1252* 1278* 1299* 1320* 1366* 1378* 1390* 1402* 1414* 1426* 1428 1440* 1442*
1455* 1457* 1488 1550* 1552* 1784* 1786* 1836* 1838* 2016* 2046* 2176* 2343* 2359* 2413* 2417
2453* 2455* 2477* 2479* 2498* 2515* 2528* 2530* 2611* 2620* 2645* 2675* 2738* 2762* 2770* 2778*
2786* 2794* 2819* 2828* 2844* 3020*
oPopPos 134 167* 219* 226* 714* 800* 979* 1157* 1168* 1233* 1545* 1558* 1804* 2193* 2225* 2277*
2285* 2426* 2457* 2481* 2500* 2541*
oPopPosAndMark 133 206* 243* 260* 278* 280* 298* 300* 315* 323* 372 389* 401 419* 444* 445*
470* 517* 519* 542 692 745 779* 801* 815* 852* 863* 893 918* 925* 945* 965*
987* 1007* 1034* 1048* 1072* 1148* 1221* 1300* 1321* 1334* 1429 1459* 1484* 1564 1623 1638
1653 1764 1789 1809* 1844* 1849* 1881 1907* 1908* 1947* 1949* 1954 1977* 1979* 1984 1994
2054* 2158 2179 2196 2214* 2228 2245* 2248 2260* 2296* 2320* 2349* 2362 2382* 2419* 2427*
2553* 2623* 2627* 2649* 2658* 2679* 2701* 2723* 2744* 2825 2848* 2852* 2862* 2875* 2885* 2923*
2936* 2971 3011* 3022* 3050 3102
oPushConstPos 140 152
oPushCopyMark 142 155* 379* 508* 718* 776* 786* 822* 1013 1023* 1082* 1103* 1295* 1310* 1316* 1328
1446* 1466* 1771* 1888* 2165* 2255* 2291* 2294* 2303 2376* 2544* 2634* 2656* 2665* 2686* 2708*

2741* 2751* 2859* 2869* 2882* 2892* 2930* 2982*

oPushCopyPos 141 154* 231 240* 354* 378 507 717* 753 775 785 821 940* 1080 1099 1105*
1109 1115* 1151 1152* 1162* 1165* 1255 1258* 1263 1266* 1285 1288* 1294 1308 1315 1362*
1374* 1386* 1398* 1410* 1422* 1445 1465 1770 1799* 1806* 1820* 1846* 1861* 1887 2005* 2254
2290 2293* 2309* 2312* 2345* 2356* 2375 2423* 2536* 2550* 2633 2655 2664 2685 2707 2740
2750 2858 2868 2872* 2881 2891 2929 2933*

oPushCurrentTokenMark 131 153* 158* 180 199 237* 253* 267* 269* 270* 287* 289* 290* 309* 330 333*
353* 356* 381* 407 424 452 456* 488 506 528 534* 702* 708 726* 734 754*
755* 825* 839* 845 846 900 904* 936 937 949 957 958 982* 994 1027* 1056
1060* 1063 1067* 1081* 1087 1091* 1100* 1110* 1119 1135* 1139 1160* 1171 1180 1197* 1208
1213* 1223 1235 1246* 1272 1293 1309* 1314 1342 1351 1356 1357 1368 1369 1380 1381
1392 1393 1404 1405 1416 1417 1433 1444 1449* 1470 1478* 1479* 1490 1536* 1555* 1775
1778* 1819* 1825 1830* 1857 1858 1900* 1933 1938 1939 1963 1968 1969 2002 2018 2023
2170 2171 2204 2205 2236 2237 2307* 2328 2336* 2352 2353 2389 2399 2400 2407 2408
2442* 2447* 2466* 2471* 2490* 2507 2517 2522* 2605* 2614* 2639 2669 2731 2756 2757 2764
2765 2772 2773 2780 2781 2788 2789 2796 2800 2805 2806 2813 2814 2830 2831 2838
2839 3001 3002 3013

oPushGcurPosNc 130 271* 273* 291* 293* 334* 382* 455* 705 710 728* 736* 757* 829* 834* 843
844 854 956 960 1030* 1090* 1095* 1248* 1274* 1327 1359* 1371* 1383* 1395* 1407* 1419*
1436 1451* 1481* 1539* 1546* 1891* 1903* 1942* 1972* 1991 2173* 2187 2190* 2209* 2223* 2240*
2302 2331* 2339* 2402 2403 2410* 2449* 2473* 2494* 2511* 2524* 2532 2607* 2616* 2641* 2671*
2734* 2759* 2767* 2775* 2783* 2791* 2802 2808 2809 2816* 2833* 2834* 2841* 3003 3006* 3017*

oPushIndentPos 139 160* 200 203* 214 223 236 251 257* 265* 285 308 312* 352* 441 499
511* 513* 533* 789* 796* 905* 938 977 981 1014* 1022 1028* 1059* 1066* 1122* 1134
1140 1143* 1159 1176* 1188 1192 1201* 1212* 1217* 1227* 1229* 1241 1345* 1471 1477 1537*
1780* 1796 1816 1832* 1895* 1901* 1934 1940 1964 1970 2024 2164 2206 2220 2238 2271
2281 2337* 2390 2435 2459 2483 2502 2543 2981*

oPushNextTokenMark 132 202* 232* 239* 256* 311* 383* 500* 514* 535* 729* 737* 797* 830* 835* 855
1031* 1096* 1106* 1113* 1116* 1125* 1145* 1153* 1155* 1163* 1166* 1177* 1189* 1193* 1202* 1218*
1230* 1242* 1249* 1256* 1259* 1264* 1267* 1275* 1286* 1289* 1346* 1363* 1375* 1387* 1399* 1411*
1423* 1437 1452* 1540* 1547* 1781* 1800* 1807* 1821* 1833* 1847* 1892* 1896* 1904* 1992 2006*
2191* 2222* 2313* 2332* 2340* 2346* 2424* 2450* 2474* 2495* 2512* 2525* 2533 2537* 2551* 2608*
2617* 2642* 2672* 2735* 2873* 2934* 3016*

oSetupGcurPosNc 129 159* 217* 255* 272* 292* 386* 433 435 437 495* 706 709 793* 795* 847
909 911 913 922* 953 959 1018 1124* 1130* 1144* 1175* 1185 1187 1200* 1216* 1240
1254 1262 1270 1280 1282 1284 1434 1474* 1634 1636 1649 1651 1779* 1828* 1842 2012
2042 2275* 2319* 2381* 2395 2401 2439* 2445* 2448* 2463* 2469* 2472* 2487* 2493* 2510* 2520*
2523* 2801 2807

OSetupPos 442*

oSetupPos 143 166* 176 201* 209* 215* 221* 224* 228* 238* 246* 252* 263* 266* 276* 283*
286* 296* 303* 310* 318* 331 350 355* 397 410 412 414 427 453 478 482
485 489 493 512* 531 700 713 716 727* 735 756* 790* 809 811 828 833
840* 850 901 906* 916 920* 932 939* 943* 950 963 972* 978* 995 1029* 1057
1064 1083* 1088 1094* 1101* 1111* 1120 1128 1136* 1141* 1161* 1172 1181 1198* 1209 1214*
1224 1228* 1236 1247* 1273 1311* 1332* 1343 1352 1358 1370 1382 1394 1406 1418 1450*
1472* 1480* 1491 1538* 1544* 1556* 1561* 1630 1645 1776 1797* 1805* 1817* 1826 1831* 1859
1902* 1935* 1941* 1945* 1952* 1957 1965* 1971* 1975* 1982* 1987 2003 2009* 2019 2025* 2039
2048 2051 2172 2188 2207* 2212* 2221* 2239* 2243* 2259* 2272* 2279* 2282* 2287* 2308* 2317*
2329 2338* 2354 2379* 2391* 2409 2422* 2436* 2443* 2460* 2467* 2484* 2491* 2503* 2508 2518
2549* 2606* 2615* 2625* 2640 2670 2732 2758 2766 2774 2782 2790 2797 2815 2832 2840
2979 2986 2989 2992 2995 2998 3004 3009* 3014

-P-

ParmOps 137*

Parser 6*

PartialSubSum 2874* 2883* 2889*

```
 PartialSum              1364* 1376* 1388* 1400* 1412* 1424* 2538* 2879*
 PartialTerm             2893* 2935* 2940*
 Pervasive                156*  380*  393* 1015* 1889*
 ProcedureDeclaration    1061* 1793*
 Programs                 148*  150*
 Project                    4*

-R-
 R                          7     8
 Reader                     8
 RecordBody              1104* 1257* 1463*
 RecordTrailer           1107* 1156* 1260* 1627*
 Relation                2736* 2742* 2748*
 RenameVariable           758*  777*  783*
 ReturnStatement         2200*
 RoutineBody             1993  1998*
 RoutineDefinition        241*  313* 1808* 1848* 1912*
 RoutineFormal           1862* 1885*

-S-
 SimpleOps                127*
 Statements              2058* 2295*
 StructureConstant        848   867*
 SubExpression           2657* 2662* 3007*
 Subscripts              3054*
 SubSum                  2752* 2760* 2768* 2776* 2784* 2792* 2803  2860* 2866*
 Sum                     1438  1447* 1453* 2411* 2545* 2643* 2810  2817* 2835* 2842* 2856*

-T-
 Tables                     6*
 tAbstraction              54
 tAligned                  55
 tAll                      56
 tAnd                      57
 tAny                      58
 tArray                    59
 tAssert                   60
 tAssign                   48
 tAsterisk                 37
 tAt                       61
 tBegin                    62
 tBind                     63
 tBits                     64
 tBlankline                65
 tBound                    66
 tCase                     67
 tChar                     22
 tCheckable                68
 tChecked                  69
 tCode                     70
 tCodeEnd                  31
 tCodeMiddle               30
 tCodeStart                29
 tCollection               71
 tColon                    51
 tComma                    49
```

```
tComment1                25
tComment2                26
tComment3                27
tComment4                28
tConst                   72
tConverter               73
tCounted                 74
tDecreasing              75
tDefault                 76
tDependent               77
tDiv                     78
tDoublePeriod            47
tDummy                   79
tElse                    80
tElself                  81
tEnd                     82
tEndOfFile               34
tEqual                   38
Term                   1360* 1372* 1384* 2927*
tExChar                  23
tExit                    83
tExports                 84
tExString                20
tExternal                85
tFinally                 86
tFor                     87
tForward                 88
tFrom                    89
tFunction                90
tGreaterEqual            42
tGreaterThan             40
tHexNumber               18
ThusItem               357*  423*
tIdent                   15
tIf                      91
tIllegal                 32
tImplies                 43
tImports                 92
tIn                      93
tInclude                 94
tInitially               95
tInline                  96
tInvariant               97
tLabel                   53
tLeftParen               44
tLessEqual               41
tLessThan                39
tLoop                    98
tMachine                 99
tMDChar                  24
tMDString                21
tMinus                   36
tMod                    100
tModule                 101
tNewLine                 33
tNot                    102
```

```
 tNumber                  16
 tOctalNumber             17
 tOf                     103
 tOr                     104
 tOtherwise              105
 tPacked                 106
 tParameter              107
 tPeriod                  46
 tPervasive              108
 tPlus                    35
 tPost                   109
 tPre                    110
 tProcedure              111
 tReadonly               112
 tRecord                 113
 tReturn                 114
 tReturns                115
 tRightParen              45
 tSemicolon               50
 tSet                    116
 tString                  19
 tThen                   117
 tThus                   118
 tTo                     119
 tType                   120
 tUnknown                121
 tUpArrow                 52
 tVar                    122
 tWhen                   123
 tWith                   124
 tXor                    125
 TypeDeclaration         161*  929*
 TypeDefinition          730*  836*  983* 1076* 1097* 1250* 1772* 1834* 1905*
 TypeFormalList          941* 1011*

-V-
 V1                        6*
 VarBindingCondition     787*  805*
 Variable                798* 1231* 1347* 1782* 2166* 2451* 2475* 2496* 2513* 2526* 2534  2983* 3026*
 VariableBinding         749*
 VariableDeclaration     696*

-W-
 WithClause              501*  515*  527*

end xref  211 identifiers  1568 total references
 291 collisions.
```

EUCLID PRETTY-PRINTER USING PASCAL

by

WUN-JEN LIN

BACHELOR OF BUSINESS, FENG-CHIA COLLEGE, 1976

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

1983

ABSTRACT

Pretty-printer is a program which is designed to format the text of a program for a particular language. It parses and reformates the input program, and generates a pretty-printer version of it. Generally speaking, it serves two purposes: one is to save a lot of time in editing the text of a program; the other is to enforce standard indentation conventions in terms of the format of a program.

S/SL (Syntax/Semantic Language) involved in this project is used as a design tool for constructing the Euclid compiler. It uses a text notation that scans syntax diagrams and adds some output and semantic routine calls. Actually, S/SL is a pure control language without data or assignment and data can be manipulated only via semantic operations. The implementation of S/SL 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 Euclid Pretty-Printer (PPP). Later, a program called S-code interpreter written in Pascal will walk through the S/SL table and execute the instructions of the S/SL program.

This report is the part of a project which is designed to make both compilation and execution of Euclid programs available on Unix 8/32 machines located in the Department of Computer Science, Kansas State University. The entire process of producing a Euclid Pretty-Printer involves the following steps :

1. The use of the S/SL program for parsing and reformatting the text of an input Euclid program.

2. The insertions and the modifications of the existing semantic routines in Pascal Pretty-Printer (PPP).

3. The modification of the lexical scanner in PPP for EPP.

4. The translation of the S/SL program into an S/SL table via an S/SL assembler and the inclusion of this table in EPP.

5. The modification of the existing S-code interpreter written in Euclid into a new version of S-code interpreter written in Pascal and the inclusion of this interpreter in EPP to access to the S/SL table.

6. The compilation and the execution of EPP.

The result generated by the above process will be an Euclid Pretty-Printer program written in Pascal. The future work of this project could be to write a PPP using S/SL, or furthermore, to rewrite the Pascal portion in EPP using Euclid.