CONCRETE SYNTAX OF PL/I

K. Alber
P. Oliva
G. Urschler
CONCRETE SYNTAX OF PL/I

by

K. ALBER
P. OLIVA
G. URSCHLER

ABSTRACT

This report supplements the semantical definition of PL/I given in "Abstract Syntax and Interpretation of PL/I" and the specification of abstract syntax given in "Translation of PL/I into Abstract Syntax" (IBM Laboratory Vienna, TR 25.082 and TR 25.086) by a syntactical definition. The syntactical form of concrete PL/I program text is defined by means of an extended Backus notation, which is described by a meta syntax.

Locator Terms for IBM Subject Index

PL/I
Backus Notation
Formal Definition
Syntax, concrete
21 PROGRAMMING

TR 25.084
28 June 1968
PREFACE

This document is part of a series of documents which represent the formal definition of syntax and semantics of PL/I issued by 28 June 1968:

/1/ LUCAS, P., LAUER, P., STIGLEITNER, H.: Method and Notation for the Formal Definition of Programming Languages.-

/2/ FLECK, M., NEUHOLD, E.: Formal Definition of the PL/I Compile Time Facilities.-

/3/ ALBER, K., OLIVA, P., URSCHLER, G.: Concrete Syntax of PL/I.-
IBM Laboratory Vienna, Techn. Report TR 25.084.

/4/ ALBER, K., OLIVA, P.: Translation of PL/I into Abstract Text.-

/5/ LUCAS, P., ALBER, K., BANDAT, K., BEKIČ, H., OLIVA, P., WALK, K., ZEISEL, G.: Informal Introduction to the Abstract Syntax and Interpretation of PL/I.-

/6/ WALK, K., ALBER, K., BANDAT, K., BEKIČ, H., CHROUST, G., KUDIELKA, V., OLIVA, P., ZEISEL, G.: Abstract Syntax and Interpretation of PL/I.-

The method and notation for these documents are essentially taken over from the first version of a formal definition of PL/I issued by the Vienna Laboratory:

/7/ PL/I Definition Group of the Vienna Laboratory: Formal Definition of PL/I.-

/8/ ALBER, K.: Syntactical Description of PL/I Text and its Translation into Abstract Normal Form.-
An outline of the method is given in /1/, which document also contains the appropriate references to the relevant literature. The basic ideas and their application to PL/I have been made available through several workshops on the formal definition of PL/I, and presentations inside and outside IBM.

The language defined in this present version is PL/I as specified in the official PL/I Language Specifications Form No. Y33-6003 with the exception of the following features which are not included:

- optimizing attributes (they are included in the concrete syntax but not in the abstract syntax; they are only tested for compatibility with other attributes and used for implication of default attributes),
- implicit conversion between offsets and pointers,
- the REFER option,
- the implicit rules for ordering initializing actions in the prologues of blocks and procedures.

The draft for this document was completed by 15 March 1968. It has been subject to validation by members of the PL/I Language Department of IBM UK Laboratories Hursley, England. The results of the checking effort conducted in Hursley have been taken into account in this present corrected form.

The formal definition given here includes more details than are given in the Specifications. These details have been confirmed as far as possible by the PL/I Language Department Hursley during the validation process. Some amendments and clarifications to the Specifications were generated during this process and will be published as Technical News Letters to the Specifications.

Contribution to the document:

Authors: their main contributions are given by chapters
- K. Alber, P. Oliva
- G. Urschler
- G. Urschler

Coordination of production: F. Schwarzenberger, H. Hoja, W. Pachl
Production of the cross-reference index: K.F. Koch
Typing: H. Deim, W. Schatzl
Special graphics: G. Lehmayer

Validation: P. Seaman, R.W. Thomas
# CONTENTS

1. INTRODUCTION 1-1

2. SYNTAX NOTATION 2-1
   2.1 Semantics of the Extended Backus Notation 2-1
   2.2 The Meta Syntax of the Extended Backus Notation 2-3
   2.3 Generation of a Concrete Program Text 2-4
      2.3.1 The Normal Generation Process 2-4
      2.3.2 Auxiliary Rules for Additional Facilities 2-5
         2.3.2.1 Keywords Abbreviations 2-6
         2.3.2.2 Multiple Closure of Blocks and Groups 2-7
      2.3.3 Programs in the 48 Character Set 2-8

3. CONCRETE SYNTAX 3-1
   3.1 Higher Level Production Rules 3-1
      3.1.1 Declarations 3-1
         3.1.1.1 Attributes 3-2
         3.1.1.2 Formats 3-5
      3.1.2 Statements 3-6
         3.1.2.1 Block and Groups 3-7
         3.1.2.2 Flow of Control Statements 3-8
         3.1.2.3 Storage Manipulating Statements 3-9
         3.1.2.4 Condition Handling Statements 3-10
         3.1.2.5 Input and Output Statements 3-11
      3.1.3 Expressions 3-12
   3.2 Lower Level Production Rules 3-14
      3.2.1 Identifiers and Constants 3-14
      3.2.2 Pictures 3-15
      3.2.3 Blanks and Comments 3-15
   3.3 List of PL/I Words 3-17
   3.4 Cross Reference Index 3-18
1. INTRODUCTION

This document is a supplement to /4/ and /6/ and contains the complete syntactical description of concrete PL/I text. It is intended for readers interested in syntactical questions who are familiar with the syntax notation of the Backus Normal Form (as used in the Revised Report on the Algorithmic Language ALGOL 60). Readers interested in the translation of concrete PL/I programs into abstract programs who are familiar with the concept of abstract objects, are referred to /4/.

The concrete syntax of PL/I is given by a set of formal production rules for writing PL/I program text in a 60-character alphabet. These rules are written in an extended Backus notation. The syntactic form and the meaning of this extended Backus notation is given in chapter 2, the production rules are listed in chapter 3.

Compile time facilities /2/ are not included in this paper. That means that not a program containing compile time macros, but a text possibly produced by them, is considered.

The concrete syntax presented in chapter 3 is a revision of the concrete syntax given in /8/. The relatively short form is a result of nearly two years' cooperation of Hursley and Vienna Laboratories.
INTRODUCTION

This academic paper is focused on the analysis and comparison of different technologies and methodologies in the field of [specific area]. The primary goal is to [explanation of the goal].

The introduction to the paper covers the background information and the objectives of the study. It highlights the significance of the research and the potential contributions to the field.

[Body of the paper]

The methodology section details the approach used in the study. This includes the selection of methods, tools, and techniques employed to achieve the research objectives.

[Results section]

The results section presents the findings of the study. This includes data analysis, comparisons, and the implications of the results.

[Conclusion section]

The conclusion summarizes the key findings and discusses their significance. It also outlines the implications for future research and practical applications.

[References section]

The references section lists all the sources and materials used in the research. This includes books, articles, and other relevant resources.
2. SYNTAX NOTATION

2.1 Semantics of the Extended Backus Notation

The attempt to give a clear and readable description of the concrete syntax of PL/I has been accompanied by the search for new notation possibilities. The extended Backus notation, which has been proved to meet all demands, is a short form of the well-known Backus notation.

In the following the meaning of the extended forms is explained by giving the equivalent forms in Backus notation.

Point of departure shall be a grammar, whose production rules have the general form:

\[ V ::= S_1 | S_2 | \ldots | S_n \]

('V' is to be replaced by one of the alternatives 'S_1' or 'S_2' or ... or 'S_n')

Note: In this chapter with 'V' variables, with 'S_i' arbitrary strings and with 'T_j' strings different from the null-string are denoted. Each of these strings may consist of a certain number of not nearer specified syntactical units, denoted by 'U_r'.

Then the introduction of the metalinguistic signs '{', '}', '[', ']', '•' (the last is a fat dot) is determined according to the following definitions:

(i) \[ V ::= S_1 T_1 S_2 \mid S_1 T_2 S_2 \mid \ldots \mid S_1 T_n S_2 \]

may be replaced by \[ V ::= S_1 \{ T_1 \mid T_2 \mid \ldots \mid T_n \} S_2 \] and vice versa

(note, that this rule remains, valid also for the case \( n = 1 \))

Example: goto-statement ::= \{ GOTO | GO TO \} reference ;

instead of: goto-statement ::= GOTO reference ; | GO TO reference ;
(ii) 'V ::= S₁ S₂ | S₁ T₁ S₂ | ... | S₁ Tₙ S₂'

may be replaced by

'V ::= S₁ [ T₁ | ... | Tₙ ] S₂' and vice versa.

Example: return-statement ::= RETURN [ ( expression ) ]

instead of: return-statement ::= RETURN ; | RETURN ( expression ) ;

(iii) 'V ::= U | V U' or

'V ::= U | U V'

may be replaced by

'V ::= U•••' and vice versa.

Note: For the inversion it is to be regarded that if 'U•••' occurs in the grammar and no production rule has the form 'V ::= U•••', this missing rule is to be added to the grammar with a not yet used variable 'V' before the replacement can be performed.

Example: integer ::= digit•••

instead of: integer ::= digit | integer digit

(iv) 'V ::= S₁ T₁ [ { T₂ T₁ } ••• ] S₂' may be replaced by

'V ::= S₁ { T₂ • T₁••• } S₂' and vice versa.

Note: Instead of 'S₁ [ { T₂ • T₁••• } ] S₂' also

'S₁ [ T₂ • T₁••• ] S₂' may be written.

Example: declarationlist ::= { , • declaration••• }

instead of: declarationlist ::= declaration [ { , declaration }•••
2.2 The Meta Syntax of the Extended Backus Notation

The expressions 'syntactical unit' and 'string of syntactical units' (called 'unit' and 'sequence' respectively in the following) have not been specified in the last section. We now define them together with the general form of the production rules of the concrete PL/I syntax recursively by means of a meta syntax. The meta syntax itself is written in the well-known Backus normal form.

To uphold the Backus notation also formally we use the metalinguistic signs '<', '>', '::=', '<', '>' in the meta syntax. Therefore we are obliged to change the notation for similar syntactic or PL/I signs. We adapt the same convention as in chapter 3, that each ambiguous sign is marked on the lower syntactic level by a further underlining. This signifies, for this and only for this sub-chapter, that the PL/I signs for colon, equal and or-sign get the forms ':', '=', '||' while the or-sign of the PL/I production rules is denoted by '||'.

Meta Syntax

<prod-rule> ::= <not-var> ::= <definition>
<definition> ::= <sequence> u <definition> | <sequence>
<sequence> ::= <unit> u <sequence> | <unit>
<unit> ::= <not-var> | <not-const> |
        { <definition> } | [ <definition> ] | <unit> >
        { <not-const> u <unit> } | [ <not-const> u <unit> ]
<not-var> ::= <sm-letter> <not-var> | <sm-letter>
<sm-letter> ::= a | b | c | d | e | f | g | h | i | j | k | l | m |
             n | o | p | q | r | s | t | u | v | w | x | y | z
<not-const> ::= <PL/I-symb> <not-const> | <PL/I-symb>
<PL/I-symb> ::= A | B | C | D | E | F | G | H | I | J | K | L | M |
             N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
             $ | @ | # | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
             blank | = | = | + | - | * | / | ( | ) | , | . | ; | ": | & | | | | | > | < | ? | % | !
Note: Each 'u' occurring in the above rules stands for one or more empty spaces (it is the sign ' ', which is not to be mixed up with the PL/I sign 'blank') in the described production rules. The appearance of 'u' in the production rule <sequence> is essential for the unambiguity of the meta-syntact. All other occurrences give only a certain freedom of formatting the production rules of the concrete PL/I syntax.

2.3 Generation of a Concrete Program Text

2.3.1 The normal generation process

First of all any implementation must provide production rules for the three implementation dependent notation variables external-option, io-option, and extralingual-character. Since PL/I has context dependent rules for the insertion of blanks and comments, which cannot be expressed by production rules of the form described in 2.1 and 2.2, the generation of a concrete PL/I program text has to be performed in four steps:

1. Starting with the notation variable "program", replacements are to be performed according to the higher level production rules listed in 3.1. This process is to be continued as long as any higher level production rule is applicable.

It ends up with a text consisting of "PL/I words", which are listed in 3.3. In this respect, all those sequences of PL/I symbols which in the production rules are not separated by empty space are assumed to compose words (notation constants) and not to be split up into their single symbols. So a word is one of the following:

- a single PL/I symbol,
- a keyword, which is a sequence of upper case letters,
- one of the eight composite operators:
  - ** || >= <= > < ->,
- one of the eight notation variables
  - identifier, simple-string-constant,
  - integer, sterling-constant,
  - isub, picture-specification,
  - real-constant,
  - imaginary-constant,
2. Now "spaces" are inserted into the generated text according to the following rule:

The 24 words

= + - * / ( ) , . ; : & | ¬ < > ** | | >= <= -, >

are "delimiters", all other words "non-delimiters". Between two adjacent non-delimiters the notation variable "space" must be inserted, between other combinations of words or following the last word of the complete program the notation variable "space" may be inserted.

The production rules for "space" are listed in 3.2.3.

3. Now the replacement is continued by application of the lower level production rules listed in 3.2.

4. Finally all notation constants are split up into their single symbols.

The complete process ends up with a text consisting of the symbols of the 60 character PL/I alphabet, i.e.

A B C D E F G H I J K L M N O P Q R S T U V W X Y
2 @ # 0 1 2 3 4 5 6 7 8 9 _ blank = + - * / ( ) , . ; : ' & | ¬ > < ? %

and extralingual characters.

2.3.2 Auxiliary rules for additional facilities

PL/I contains two facilities which in the one case would lengthen unnecessarily the production rules and in the other case cannot be expressed by context independent production rules. Both facilities allow a program text to be replaced by a shorter one, without changing the semantical meaning.
2.3.2.1 Keyword abbreviations

The following abbreviations may be inserted instead of the corresponding keywords. This replacement has to be performed before step 3 of the generation process described in 2.3 is performed:

**keywords:**
- ABNORMAL
- AUTOMATIC
- BINARY
- BUFFERED
- CHARACTER
- COLUMN
- COMPLEX
- CONTROLLED
- CONVERSION
- DECIMAL
- DECLARE
- DEFINED
- ENVIRONMENT
- EXCLUSIVE
- EXTERNAL
- FIXEDOVERFLOW
- INITIAL
- INTERNAL
- IRREDUCIBLE
- NOCONVERSION
- NOFIXEDOVERFLOW
- NOOVERFLOW
- NOSTRINGRANGE
- NOUNDERFLOW
- NOZEROODIVIDE
- OVERFLOW
- PICTURE
- POINTER
- POSITION
- PROCEDURE
- REDUCIBLE
- SEQUENTIAL
- STRINGRANGE

**abbreviations:**
- ABNL
- AUTO
- BIN
- BUF
- CHAR
- COL
- CPLX
- CTL
- CON
- DEC
- DCL
- DEF
- ENV
- EXCL
- EXT
- FOFL
- INIT
- INT
- IRRED
- NOCONV
- NOFOFL
- NOOFL
- NOSTRG
- NOSUBRG
- NOUFL
- NOZDIV
- OPL
- PIC
- PTR
- POS
- PROC
- RED
- QEL
- STRG
2.3.2.2 Multiple closure of blocks and groups

Assume, that all four steps of the generation process described in 2.3.1 including the insertion of abbreviated keywords have been terminated.

Then a part of this program text is called a compound if it could have been generated by means of the following production rules:

\[
\text{compound} :: \text{compoundhead} \ [ \text{compound} \ ] \ \text{compoundend}
\]

\[
\text{compoundhead} :: \ [ \text{prefixlist} \ ]
\]

\[
\{ \ [ \text{labellist} \] \ \text{BEGIN} \ [ \text{OPTIONS} \ \{ \ , \ . \ \text{external-option} \cdots \} \ ] \ [ \text{entry-namelist} \ \{ \ PROC \ | \ PROCEDURE \} \ [ \text{parameterlist} \ [ \text{procedure-optionslist} \]
\]

\[
\{ \ [ \text{labellist} \] \ \text{DO} \ [ \text{do-specification} | \ \text{WHILE} \ ( \text{expression} ) \ ] \}
\]

\[
\[ \text{sentence} \cdots \]
\]

\[
\text{compoundend} :: \ [ \text{prefixlist} \ ] \ [ \text{labellist} \] \ \text{END} ;
\]

Before the rightmost semicolon of a compound, i.e. between 'END' and ';', one identifier of the labellist or entry-namelist of the appertaining compoundhead may be inserted.

Provided that a compound actually has such a compoundend, e.g. 'END identifier_1 ;' it is allowed to omit an immediately preceding compoundend with no prefix- or labellist (i.e. a compoundend immediately to the left) if identifier_1 does not occur in the labellist or entry-namelist of the compoundhead appertaining to the omitted compoundend.
2.3.3 Programs in the 48 character set

It is possible to write PL/I programs in the following 48 character set:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9 blank = + - * / ( )
```

If the program shall be written in this character set, in addition to the processes described in 2.3.1 and 2.3.2 the following rules have to be obeyed:

1) From the production rules for "letter", "alphanumeric-character", "string-character" and "comment-symbol" the following 12 symbols have to be deleted:

```
@ # $ % & * ( ) [ ] ^ _ \ ~
```

2) The following 13 PL/I words have to be handled as notation variables and to be replaced by means of the (higher level) production rules:

```
; ::= , . ::= GE
- ::= NOT <= ::= LE
& ::= AND -> ::= NG
| ::= OR <- ::= NL
> ::= GT = ::= NE
< ::= LT || ::= CAT
```

For the insertion of spaces the word '..' is handled as a delimiter and the other 12 words resulting from these replacements as non-delimiters.

3) The 12 sequences of letters

```
NOT, AND, OR, GT, LT, GE, LE, NG, NL, NE, CAT, PT
```

are "reserved words", i.e. no identifier must finally be replaced by any of these sequences.

4) In the final text, each colon ' :' is to be replaced:
   a) when immediately following a dot ' .' by means of the production rules

```
:: = space ..
```

b) else by means of the production rule

```
; ::= 
```
3. CONCRETE SYNTAX

3.1 Higher level production rules

(1) program ::= procedure

(2) procedure ::= [ prefixlist ] entry-namelist PROCEDURE [ parameter-list ] [ procedure-optionslist ] ; sentence-list

(3) entry-namelist ::= ( identifier )

(4) parameter-list ::= ( , * identifier )

(5) procedure-optionslist ::= { function-attribute | RECURSIVE | OPTIONS ( , * external-option ) } sentence-list

(6) sentence-list ::= [ sentence ] end-clause

(7) end-clause ::= [ prefixlist ] [ label-list ] END ;

(8) sentence ::= statement | procedure | entry | declaration-sentence | format-sentence

3.1.1 Declarations:

(9) declaration-sentence ::= [ label-list ] DECLARE declaration-list ;
(10) declarationlist ::=  
{ · · declaration···· }  

(11) declaration ::=  
[ integer ] [ identifier ] ( declarationlist )  
[ dimension-attribute ] [ attribute···· ]  

(12) dimension-attribute ::=  
( · · · bound-pair···· ) }  

(13) bound-pair ::=  
[ expression : ] expression | *  

3.1.1 Attributes:  

(14) attribute ::=  
data-attribute | non-data-attribute | scope-attribute | like-attribute  

(15) data-attribute ::=  
arithmetic-attribute | string-attribute | VARYING |  
picture-attribute | area-attribute | label-attribute  
POINTEB | offset-attribute | TASK | EVENT | CELL |  
storage-class-attribute | defined-attribute | based-attribute |  
PACKED | ALIGNED | SECONDARY | NORMAL | ABNORMAL |  
initial-attribute  

(16) arithmetic-attribute ::=  
[ REAL | COMPLEX | DECIMAL | BINARY | FLOAT |  
FIXED ] ( integer [ , signed-integer ] ) ]  

(17) signed-integer ::=  
[ + | - ] integer  

(18) string-attribute ::=  
[ BIT | CHARACTER ] ( expression * )  

(19) picture-attribute ::= 

\[ PICTURE \text{ picture-specification} \]

\[
\begin{align*}
\text{(16) picture-specification} &\quad \text{[\body]}
\end{align*}
\]

(20) area-attribute ::= 

\[ AREA \{ \text{expression} \ast \} \]

(21) label-attribute ::= 

\[ LABEL \{ \text{identifier} \ast \} \]

(22) offset-attribute ::= 

\[ OFFSET \{ \text{reference} \} \]

(23) storage-class-attribute ::= 

\[ AUTOMATIC | STATIC | CONTROLLED \]

(24) defined-attribute ::= 

\[ DEFINED \text{ basic-reference} | \text{POSITION} \{ \text{integer} \} \]

(25) based-attribute ::= 

\[ BASED \{ \text{reference} \} \]

(26) initial-attribute ::= 

\[ INITIAL \{ \text{initial-call} | \text{initial-itemlist} \} \]

(27) initial-call ::= 

\[ CALL \text{ identifier} \{ \text{argumentlist} \} \]

(28) initial-itemlist ::= 

\[ \{ \text{initial-item} \ast \} \]

(29) initial-item ::= 

\[ \text{initial-iteration} | \text{initial-constant} | \text{simple-string-constant} | \ast \]

(30) initial-iteration ::= 

\[ \{ \text{expression} \} \{ \text{initial-constant} | \text{initial-itemlist} \} \]

(31) initial-constant ::= 

\[ \text{identifier} | \text{replicated-string-constant} | \text{arithmetic-initial-constant} | \text{sterling-constant} \]
(32) arithmetic-initial-constant ::= 
[ + | - ] real-constant 
| [ + | - ] imaginary-constant 

(33) non-data-attribute ::= 
entry-name-attribute | file-name-attribute 
BUILTIN | generic-attribute

(34) entry-name-attribute ::= 
ENTRY [ ( descriptorlist ) ] 
RETURNS ( function-attribute*** ) 
USES | SETS ) ( ( , * uses-sets-item*** ) 
REDUCIBLE | IRREDUCIBLE

(35) descriptorlist ::= 
descriptor [ , descriptorlist]

(36) descriptor ::= 
[ integer ][ dimension-attribute ][ attribute*** ]

(37) function-attribute ::= 
arithmetic-attribute | string-attribute | VARYING | ATTACH 
picture-attribute | area-attribute | POINTER | offset-attribute

(38) uses-sets-item ::= 
unsubscripted-reference | integer | *

(39) file-name-attribute ::= 
FILE | file-attribute | ENVIRONMENT ( io-option )

(40) file-attribute ::= 
STREAM | RECORD | INPUT | OUTPUT | UPDATE | 
SEQUENTIAL | DIRECT | BUFFERED | UNBUFFERED | 
KEYED | PRINT | BACKWARDS | EXCLUSIVE

(41) generic-attribute ::= 
GENERIC ( declarationlist )
3.1.1.2 Formats:

(44) format-sentence ::= [ prefixlist ] labellist FORMAT formatlist;

(45) formatlist ::= 

(46) format ::= format-iteration | format-item

(47) format-iteration ::= 

(48) format-item ::= 

(49) data-format | control-format | remote-format

(50) real-format ::= 

(51) complex-format ::= 

(52) string-format ::= [ B | A ] [ ( expression ) ]
(53) picture-format ::=
\[ \text{picture-specification} \]

(54) control-format ::= 
\[ \{ \text{COLUMN} | \text{LINE} | \text{PAGE} | \text{SKIP} | \text{X} \} [ \{ \text{expression} \} ] \]

(55) remote-format ::= 
\[ \text{R} ( \text{reference} ) \]

3.1.2 Statements:

(56) entry ::= 
\[ \text{entry-namelist ENTRY [ parameterlist [ attribute••• ] ]} ; \]

(57) statement ::= 
\[ \text{if-statement | unconditional-statement} \]

(58) unconditional-statement ::= 
\[ \text{begin-block | simple-statement} \]

(59) simple-statement ::= 
\[ [ \text{prefixlist} ] [ \text{labellist} ] \text{proper-statement} \]

(60) prefixlist ::= 
\[ \{ \{ \ldots \text{prefix-element•••} \} \ldots \}\]

(61) prefix-element ::= 
\[ \text{prefix | no-prefix | check-condition} \]
\[ \text{no-check-condition} \]

(62) prefix ::= 
\[ \text{CONVERSION | FIXEDOVERFLOW | OVERFLOW | SIZE} \]
\[ \text{SUBSCRIPTRANGE | STRINGRANGE | UNDERFLOW | ZEBO_DIVIDE} \]
(63) no-prefix ::=                   \(1\)  
    NOCONVERSION | NOFIXEDOVERFLOW | NOOVERFLOW |  
    NOSIZE | NOSUBSCRIPTRANGE | NOSTRINGRANGE | NOUNDERFLOW |  
    NOZERODIVIDE  

(64) labellist ::=                \(1\)  
    [ ( identifier | initial-label ) ; ]***  

(65) initial-label ::=           \(1\)  
    [ expression BY expression ]  
    [ ( identifier [ ( [ , * signed-integer** ] ) ] ) ]***  
    identifier [ ( [ , * signed-integer** ] ) ] ***  
    [ ( * identifier ) ] ***  

(66) proper-statement ::=         \(1\)  
    group | goto-statement | call-statement |  
    return-statement | wait-statement | delay-statement |  
    exit-statement | stop-statement | assignment-statement |  
    allocate-statement | free-statement | on-statement |  
    revert-statement | signal-statement | open-statement |  
    close-statement | stream-io-statement | record-io-statement |  
    display-statement | null-statement  

(67) null-statement ::= \(1\)  
    ;  

3.1.2.1 Block and groups:  

(68) begin-block ::=           \(1\)  
    [ prefixlist ] [ labellist ] BEGIN  
    [ OPTIONS ( [ , * external-option** ] ) ] ; sentencelist  

(69) group ::=                \(1\)  
    simple-group | iterated-group  
    \{ ( identifier | expression ) \} APPEND  
    \{ \} \} APPEND  

(70) simple-group ::=           \(1\)  
    DO ; sentencelist
3.1.2.2 Flow of control statements:

(71) iterated-group ::=  
    DO { do-specification | WHILE ( expression ) } 

(72) do-specification ::=  
    reference = { , * specification*** }  

(73) specification ::=  
    expression [ BY expression [ TO expression ] ]  
    TO expression [ BY expression ] * | } ) | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | } | |
3.1.2.3 Storage manipulating statements:

(86) assignment-statement ::= = expression [ , BY NAME ] ;

(87) allocate-statement ::= ALLOCATE [ , { based_allocate_item
controlled_allocate_item } ] ;

(88) based_allocate_item ::= identifier [ SET ( reference ) [ IN ( reference ) ] ]

(89) controlled_allocate_item ::= identifier [ dimension_attribute
{ string_attribute | area_attribute = CELL | initial_attribute } ]
(90) free-statement ::=  
FREE (, * ( [ reference -> ] identifier [ IN ( reference ) ] ) ) ;

3.1.2.4 Condition handling statements:

(91) on-statement ::=  
ON condition [ SNAP ] { unconditional-statement |  
SYSTEM ; }  

(92) revert-statement ::=  
REVERT condition ;

(93) signal-statement ::=  
SIGNAL condition ;

(94) condition ::=  
prefix | check-condition | AREA | named-io-condition |  
ERROR | FINISH | programmer-named-condition

(95) check-condition ::=  
CHECK ( [ , * unsubscribed-reference ] )  

(96) no-check-condition ::=  
NOCHECK ( [ , * unsubscribed-reference ] )

(97) named-io-condition ::=  
io-condition ( identifier )

(98) io-condition ::=  
ENDFILE | ENDPAGE | KEY | NAME | RECORD | TRANSMIT |  
UNDEFINEDFILE

(99) programmer-named-condition ::=  
CONDITION ( identifier )
3.1.2.5 Input and output statements:

(100) open-statement ::=  
OPEN { , * open-optionslist*** } ;

(101) open-optionslist ::=  
{ file-attribute | FILE(identifier) | IDENT(expression) |  
TITLE(expression) | LINESIZE(expression) |  
PAGESIZE(expression) }***

(102) close-statement ::=  
CLOSE { , * close-optionslist*** } ;

(103) close-optionslist ::=  
{ FILE identifier | IDENT expression }***

(104) stream-io-statement ::=  
{ GET | PUT } stream-optionslist ;

(105) stream-optionslist ::=  
{ FILE identifier | STRING reference | data-specification |  
COPY | SKIP [ expression ] | PAGE | LINE [ expression ] }***

(106) data-specification ::=  
data-directed | edit-directed | list-directed

(107) data-directed ::=  
DATA [ ( datalist ) ]

(108) edit-directed ::=  
EDIT [ ( datalist ) formatlist ]***

(109) list-directed ::=  
LIST ( datalist )

(110) datalist ::=  
{ , * datalist-element***}

(111) datalist-element ::=  
( datalist DO do-specification ) | expression
(112) record-io-statement ::=  
  { READ | WRITE | REWRITE | LOCATE | DELETE | UNLOCK }  
  [ identifier ] record-optionslist ;

(113) record-optionslist ::=  
  { FILE ( identifier ) | EVENT ( reference ) | FROM ( reference ) |  
    IGNORE ( expression ) | INTO ( reference ) | KEY ( expression ) |  
    KEYTO ( reference ) | KEYPROM ( expression ) | SET ( reference ) |  
    NOLOCK } * * *

(114) display-statement ::=  
  DISPLAY ( expression )  
  [ REPLY ( reference ) [ EVENT ( reference ) ] ]  
  EVENT ( reference ) REPLY ( reference ) ;

3.1.3 Expressions:

(115) expression ::=  
  expression-six | expression  
  expression-six

(116) expression-six ::=  
  expression-five | expression-six & expression-five

(117) expression-five ::=  
  expression-four | expression-five comparison-operator expression-four

(118) comparison-operator ::=  
  > | >= | = | < | <= | <> | <=|<> | <＞|＜|＞< |

(119) expression-four ::=  
  expression-three | expression-four || expression-three

(120) expression-three ::=  
  expression-two | expression-three [ + | - ] expression-two

(121) expression-two ::=  
  expression-one | expression-two [ * | / ] expression-one
(122) expression-one ::= 
    primitive-expression | { * | - | } expression-one |
    primitive-expression ** expression-one
(123) primitive-expression ::= 
    ( expression ) | reference | constant | isub
(124) reference ::= 
    [ reference -> ] basic-reference
(125) basic-reference ::= 
    [ unqualified-reference ]
(126) unqualified-reference ::= 
    identifier [ { expression | * } ]
(127) unsubscripted-reference ::= 
    [ identifier ]
(128) constant ::= 
    real-constant | imaginary-constant | sterling-constant |
    simple-string-constant | replicated-string-constant
(129) replicated-string-constant ::= 
    ( integer ) simple-string-constant
3.2 Lower level production rules:

3.2.1 Identifiers and constants:

(130) identifier ::= letter [ alphaner-character***]

(131) letter ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | @ | ^ |

(132) alphaner-character ::= letter | digit | _

(133) digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

(134) isub ::= integer SUB

(135) integer ::= digit***

(136) real-constant ::= [ fixed-constant | float-constant ] [ B- ] sign ( integer )

(137) fixed-constant ::= integer [ - ] [ integer ] . integer

(138) float-constant ::= fixed-constant E [ + | - ] integer

(139) imaginary-constant ::= real-constant I

(140) simple-string-constant ::= bit-string | character-string

(141) bit-string ::= ' [ bit*** ] ' B
(142) bit ::=  
    0 | 1

(143) character-string ::=  
    ' [ string-character••• ] '  

(144) string-character ::=  
    alphabetic-character | blank | '' | extralingual-character
    = | + | - | * | / | { | } | ! | . | : | & | \ | ~ | > | < | ? | % | 

(145) sterling-constant ::=  
    integer . integer . fixed-constant L

3.2.2 Pictures:

(146) picture-specification ::=  
    ' picture-string [ F ( [ + | - ] integer ) ] '  

(147) picture-string ::=  
    [ ( integer ) ] picture-character [ picture-string ]

(148) picture-character ::=  
    A | B | C | D | E | G | H | I | K | M | P | R | S | T | V | X | Y | Z | $ | 1 | 2 | 3 | 6 | 7 | 8 | 9 | + | - | * | / | \ | .

3.2.3 Blanks and comments:

(149) space ::=  
    { blank | comment }•••
(150) comment ::= 
   /* [ { [ **** ] comment-symbol | / }*** ] *** */ 0

(151) comment-symbol ::= 
   alphameric-character | blank | ' ' | character-string
   = + | - | ( | ) | , | . | ; | : | & | \ | \ | ] | ] | ] | ] | ]
   extralingual-character
   extralingual-character

3.5.1 Pictur::

(146) pictur-representation ::= [ ] pictur-representation

(147) pictur-rotation ::= [ ] pictur-rotation

(148) pictur-character ::= |

3.5.2 Picture and comment:

(149) space ::= [ ] comment { blank [ ] ]

(150) id ::= 

### 3.3 List of PL/I words:

<table>
<thead>
<tr>
<th>PL/I Word</th>
<th>PL/I Word</th>
<th>PL/I Word</th>
<th>PL/I Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPLEX</td>
<td>INITIAL</td>
<td>PUT</td>
<td></td>
</tr>
<tr>
<td>CONDITION</td>
<td>INPUT</td>
<td>READ</td>
<td></td>
</tr>
<tr>
<td>CONTROLLED</td>
<td>integer</td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>CONVERSION</td>
<td>INTERNAL</td>
<td>real-constant</td>
<td></td>
</tr>
<tr>
<td>COPY</td>
<td>INTO</td>
<td>RECORD</td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>IRREDUCIBLE</td>
<td>RECURSIVE</td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td>isub</td>
<td>REDUCIBLE</td>
<td></td>
</tr>
<tr>
<td>DECLARE</td>
<td>KEY</td>
<td>RPLPLY</td>
<td></td>
</tr>
<tr>
<td>DEFINED</td>
<td>KEYED</td>
<td>RETURN</td>
<td></td>
</tr>
<tr>
<td>DELAY</td>
<td>KEYFROM</td>
<td>RETURNS</td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td>KEYTO</td>
<td>REVERT</td>
<td></td>
</tr>
<tr>
<td>DIRECT</td>
<td>LABEL</td>
<td>REWRITE</td>
<td></td>
</tr>
<tr>
<td>DISPLAY</td>
<td>LIKE</td>
<td>SECONDARY</td>
<td></td>
</tr>
<tr>
<td>DO</td>
<td>LINE</td>
<td>SEQUENTIAL</td>
<td></td>
</tr>
<tr>
<td>EDIT</td>
<td>LINESIZE</td>
<td>SET</td>
<td></td>
</tr>
<tr>
<td>ELSE</td>
<td>LIST</td>
<td>SIMPLE-STRING-CONSTANT</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td>LOCATE</td>
<td>SIZE</td>
<td></td>
</tr>
<tr>
<td>ENDFILE</td>
<td>NAME</td>
<td>SKIP</td>
<td></td>
</tr>
<tr>
<td>ENDPAGE</td>
<td>NOCHECK</td>
<td>SNAP</td>
<td></td>
</tr>
<tr>
<td>ENTRY</td>
<td>NOCONVERSION</td>
<td>STATIC</td>
<td></td>
</tr>
<tr>
<td>ENVIRONMENT</td>
<td>NOPROPERTY</td>
<td>STERLING-CONSTANT</td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td>NOPRINT</td>
<td>STOP</td>
<td></td>
</tr>
<tr>
<td>EVENT</td>
<td>NOREAD</td>
<td>STRING</td>
<td></td>
</tr>
<tr>
<td>EXCLUSIVE</td>
<td>NOSET</td>
<td>SUBSTRING</td>
<td></td>
</tr>
<tr>
<td>EXIT</td>
<td>NOSIZE</td>
<td>SYSTEM</td>
<td></td>
</tr>
<tr>
<td>ABNORMAL</td>
<td>NOSTRINGRANGE</td>
<td>TASK</td>
<td></td>
</tr>
<tr>
<td>ALIGNED</td>
<td>NOSTRINGRANGE</td>
<td>THEN</td>
<td></td>
</tr>
<tr>
<td>ALLOCATE</td>
<td>NOSUBSCRIPTRANGE</td>
<td>TITLE</td>
<td></td>
</tr>
<tr>
<td>AREA</td>
<td>NOUNDERFLOW</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>AUTOMATIC</td>
<td>OFFSET</td>
<td>TRANSMIT</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>OPTIONS</td>
<td>UNBUFFERED</td>
<td></td>
</tr>
<tr>
<td>BACKWARDS</td>
<td>OUTPUT</td>
<td>UNDEFINEDFILE</td>
<td></td>
</tr>
<tr>
<td>BASED</td>
<td>OVERFLOW</td>
<td>UndefsFlow</td>
<td></td>
</tr>
<tr>
<td>BEGIN</td>
<td>P</td>
<td>UNLOCK</td>
<td></td>
</tr>
<tr>
<td>BIT</td>
<td>PAKED</td>
<td>USES</td>
<td></td>
</tr>
<tr>
<td>BUFFERED</td>
<td>PAGE</td>
<td>VARYING</td>
<td></td>
</tr>
<tr>
<td>BUILTIN</td>
<td>PAGESIZE</td>
<td>WAIT</td>
<td></td>
</tr>
<tr>
<td>BY</td>
<td>PICTURE</td>
<td>WHILE</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>pointer</td>
<td>WRITE</td>
<td></td>
</tr>
<tr>
<td>CALL</td>
<td>picture-specification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CELL</td>
<td>POSITION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHARACTER</td>
<td>PRINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHECK</td>
<td>pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOSE</td>
<td>priority</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLUMN</td>
<td>procedure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imaginary-constant</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Cross Reference Index:

<table>
<thead>
<tr>
<th>3-1(4)</th>
<th>3-1(5)</th>
<th>3-2(11)</th>
<th>3-2(12)</th>
<th>3-2(16)</th>
<th>3-2(18)</th>
<th>3-3(20)</th>
<th>3-3(21)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-3(22)</td>
<td>3-3(24)</td>
<td>3-3(25)</td>
<td>3-3(28)</td>
<td>3-3(30)</td>
<td>3-4(34)</td>
<td>3-4(34)</td>
<td>3-5(51)</td>
</tr>
<tr>
<td>3-6(54)</td>
<td>3-6(55)</td>
<td>3-6(60)</td>
<td>3-7(65)</td>
<td>3-7(65)</td>
<td>3-7(68)</td>
<td>3-8(71)</td>
<td>3-8(73)</td>
</tr>
<tr>
<td>3-9(82)</td>
<td>3-9(82)</td>
<td>3-9(88)</td>
<td>3-9(88)</td>
<td>3-9(88)</td>
<td>3-10(90)</td>
<td>3-10(95)</td>
<td>3-10(96)</td>
</tr>
<tr>
<td>3-11(101)</td>
<td>3-11(104)</td>
<td>3-11(113)</td>
<td>3-12(113)</td>
<td>3-12(114)</td>
<td>3-12(114)</td>
<td>3-13(126)</td>
<td></td>
</tr>
<tr>
<td>3-14(143)</td>
<td>3-14(145)</td>
<td>3-15(146)</td>
<td>3-15(147)</td>
<td>3-15(161)</td>
<td>3-15(165)</td>
<td>3-15(167)</td>
<td>3-15(167)</td>
</tr>
<tr>
<td>3-16(161)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
</tr>
<tr>
<td>3-17(17)</td>
<td>3-4(32)</td>
<td>3-4(32)</td>
<td>3-12(120)</td>
<td>3-13(122)</td>
<td>3-14(138)</td>
<td>3-15(144)</td>
<td>3-15(146)</td>
</tr>
<tr>
<td>3-16(165)</td>
<td>3-15(161)</td>
<td>3-15(161)</td>
<td>3-15(161)</td>
<td>3-15(161)</td>
<td>3-15(161)</td>
<td>3-15(161)</td>
<td>3-15(161)</td>
</tr>
<tr>
<td>3-17(17)</td>
<td>3-2(18)</td>
<td>3-3(20)</td>
<td>3-3(29)</td>
<td>3-4(38)</td>
<td>3-12(121)</td>
<td>3-13(126)</td>
<td>3-16(150)</td>
</tr>
<tr>
<td>3-15(144)</td>
<td>3-15(148)</td>
<td>3-16(150)</td>
<td>3-16(150)</td>
<td>3-16(150)</td>
<td>3-16(150)</td>
<td>3-16(150)</td>
<td>3-16(150)</td>
</tr>
<tr>
<td>3-17(17)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
</tr>
<tr>
<td>3-3(22)</td>
<td>3-3(24)</td>
<td>3-3(25)</td>
<td>3-3(28)</td>
<td>3-3(30)</td>
<td>3-4(34)</td>
<td>3-4(34)</td>
<td>3-5(51)</td>
</tr>
<tr>
<td>3-6(54)</td>
<td>3-6(55)</td>
<td>3-6(60)</td>
<td>3-7(65)</td>
<td>3-7(65)</td>
<td>3-7(68)</td>
<td>3-8(71)</td>
<td>3-8(73)</td>
</tr>
<tr>
<td>3-9(82)</td>
<td>3-9(82)</td>
<td>3-9(88)</td>
<td>3-9(88)</td>
<td>3-9(88)</td>
<td>3-10(90)</td>
<td>3-10(95)</td>
<td>3-10(96)</td>
</tr>
<tr>
<td>3-11(101)</td>
<td>3-11(104)</td>
<td>3-11(113)</td>
<td>3-12(113)</td>
<td>3-12(114)</td>
<td>3-12(114)</td>
<td>3-13(126)</td>
<td></td>
</tr>
<tr>
<td>3-14(143)</td>
<td>3-14(145)</td>
<td>3-15(146)</td>
<td>3-15(147)</td>
<td>3-15(161)</td>
<td>3-15(165)</td>
<td>3-15(167)</td>
<td>3-15(167)</td>
</tr>
<tr>
<td>3-16(161)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
</tr>
<tr>
<td>3-17(17)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
</tr>
<tr>
<td>3-3(22)</td>
<td>3-3(24)</td>
<td>3-3(25)</td>
<td>3-3(28)</td>
<td>3-3(30)</td>
<td>3-4(34)</td>
<td>3-4(34)</td>
<td>3-5(51)</td>
</tr>
<tr>
<td>3-6(54)</td>
<td>3-6(55)</td>
<td>3-6(60)</td>
<td>3-7(65)</td>
<td>3-7(65)</td>
<td>3-7(68)</td>
<td>3-8(71)</td>
<td>3-8(73)</td>
</tr>
<tr>
<td>3-9(82)</td>
<td>3-9(82)</td>
<td>3-9(88)</td>
<td>3-9(88)</td>
<td>3-9(88)</td>
<td>3-10(90)</td>
<td>3-10(95)</td>
<td>3-10(96)</td>
</tr>
<tr>
<td>3-11(101)</td>
<td>3-11(104)</td>
<td>3-11(113)</td>
<td>3-12(113)</td>
<td>3-12(114)</td>
<td>3-12(114)</td>
<td>3-13(126)</td>
<td></td>
</tr>
<tr>
<td>3-14(143)</td>
<td>3-14(145)</td>
<td>3-15(146)</td>
<td>3-15(147)</td>
<td>3-15(161)</td>
<td>3-15(165)</td>
<td>3-15(167)</td>
<td>3-15(167)</td>
</tr>
<tr>
<td>3-16(161)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
</tr>
<tr>
<td>3-17(17)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
<td>3-2(18)</td>
</tr>
<tr>
<td>3-3(22)</td>
<td>3-3(24)</td>
<td>3-3(25)</td>
<td>3-3(28)</td>
<td>3-3(30)</td>
<td>3-4(34)</td>
<td>3-4(34)</td>
<td>3-5(51)</td>
</tr>
<tr>
<td>3-6(54)</td>
<td>3-6(55)</td>
<td>3-6(60)</td>
<td>3-7(65)</td>
<td>3-7(65)</td>
<td>3-7(68)</td>
<td>3-8(71)</td>
<td>3-8(73)</td>
</tr>
<tr>
<td>3-9(82)</td>
<td>3-9(82)</td>
<td>3-9(88)</td>
<td>3-9(88)</td>
<td>3-9(88)</td>
<td>3-10(90)</td>
<td>3-10(95)</td>
<td>3-10(96)</td>
</tr>
<tr>
<td>3-11(101)</td>
<td>3-11(104)</td>
<td>3-11(113)</td>
<td>3-12(113)</td>
<td>3-12(114)</td>
<td>3-12(114)</td>
<td>3-13(126)</td>
<td></td>
</tr>
<tr>
<td>3-14(143)</td>
<td>3-14(145)</td>
<td>3-15(146)</td>
<td>3-15(147)</td>
<td>3-15(161)</td>
<td>3-15(165)</td>
<td>3-15(167)</td>
<td>3-15(167)</td>
</tr>
<tr>
<td>3-16(161)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
<td>3-16(165)</td>
</tr>
</tbody>
</table>
argumentlist 3-3(27), 3-8(78), 3-2(80)
arithmetic-attribute 3-2(15), 3-2(16), 3-4(37)
arithmetic-initial-constant 3-3(31), 3-4(32)
assignment-statement 3-7(66), 3-2(86)
attribute 3-2(11), 3-2(14), 3-4(36), 3-6(56)
AUTOMATIC 3-3(23)
B 3-5(52), 3-14(131), 3-14(136), 3-14(141), 3-15(148)
BACKWARDS 3-4(40)
balanced-statement 3-8(74), 3-8(76), 3-8(76), 3-8(76)
based-allocate-item 3-9(87), 3-9(89)
based-attribute 3-2(15), 3-3(25)
begin-block 3-2(16)
BEGIN 3-6(58), 3-7(60)
BIT 3-14(141), 3-15(142)
BIT 3-2(18)
bit-string 3-14(140), 3-14(141)
blank 3-15(144), 3-15(149), 3-16(151)
bound-pair 3-2(12), 3-2(13)
BUFFERED 3-4(40)
BUILTIN 3-4(33)
BY 3-8(73), 3-8(73), 3-9(86)
C 3-15(148), 3-5(51), 3-14(131)
CALL 3-3(27), 3-8(78)
call-optionslist 3-8(78), 3-8(79)
call-statement 3-7(66), 3-8(79)
CELL 3-2(15), 3-9(89)
CHARACTER 3-2(18)
character-string

CHECK

check-condition

CLOSE

close-optionslist

close-statement

COLUMN

comment

comment-symbol

comparison-operator

COMPLEX

complex-format

condition

CONDITION

constant

control-format

CONTROLLED

controlled-allocate-item

CONVERSION

COPY

D

DATA

data-attribute

data-directel

data-format

data-specification

data-list

data-list-element

3-14(140), 3-15(143)

3-10(95)

3-6(61), 3-10(94), 3-10(95)

3-11(102)

3-11(102), 3-11(103)

3-7(66), 3-11(102)

3-6(54)

3-15(149), 3-16(150)

3-16(150), 3-16(151)

3-12(117), 3-12(118)

3-2(16)

3-5(49), 3-5(51)

3-10(91), 3-10(92), 3-10(93), 3-10(94)

3-10(99)

3-13(123), 3-13(128)

3-5(48), 3-6(54)

3-3(23)

3-9(87), 3-9(92)

3-6(62)

3-11(105)

3-15(148), 3-14(131)

3-11(107)

3-2(14), 3-2(15)

3-11(106), 3-11(107)

3-5(48), 3-5(49)

3-11(105), 3-11(106)

3-11(107), 3-11(108), 3-11(109), 3-11(110), 3-11(111)

3-11(110), 3-11(111)
<table>
<thead>
<tr>
<th>Command</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECLAR E</td>
<td>3-2</td>
</tr>
<tr>
<td>DECLAR ATION</td>
<td>3-2</td>
</tr>
<tr>
<td>DECLAR ATION-SENTENCE</td>
<td>3-1</td>
</tr>
<tr>
<td>DECLAR ATIONLIST</td>
<td>3-1</td>
</tr>
<tr>
<td>DEF INED</td>
<td>3-2</td>
</tr>
<tr>
<td>DEF INED-ATTRIBUTE</td>
<td>3-2</td>
</tr>
<tr>
<td>DEL AY</td>
<td>3-2</td>
</tr>
<tr>
<td>DEL AY-STATEMENT</td>
<td>3-2</td>
</tr>
<tr>
<td>DELET E</td>
<td>3-2</td>
</tr>
<tr>
<td>DESCR IP T</td>
<td>3-2</td>
</tr>
<tr>
<td>DESCR IP TLIST</td>
<td>3-2</td>
</tr>
<tr>
<td>DIG I T</td>
<td>3-2</td>
</tr>
<tr>
<td>DIMENSION-ATTRIBUTE</td>
<td>3-2</td>
</tr>
<tr>
<td>DIRECT</td>
<td>3-2</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>3-2</td>
</tr>
<tr>
<td>DISPLAY-STATEMENT</td>
<td>3-2</td>
</tr>
<tr>
<td>DO</td>
<td>3-2</td>
</tr>
<tr>
<td>DO-SPECIFICATION</td>
<td>3-2</td>
</tr>
<tr>
<td>E DIT</td>
<td>3-2</td>
</tr>
<tr>
<td>EDIT-DIRECTED</td>
<td>3-2</td>
</tr>
<tr>
<td>ELSE</td>
<td>3-2</td>
</tr>
<tr>
<td>END</td>
<td>3-2</td>
</tr>
<tr>
<td>END-CLAUSE</td>
<td>3-2</td>
</tr>
<tr>
<td>ENDP FILE</td>
<td>3-2</td>
</tr>
<tr>
<td>ENDPAGE</td>
<td>3-2</td>
</tr>
<tr>
<td>ENTRY</td>
<td>3-2</td>
</tr>
<tr>
<td>ENTRY-NAME-ATTRIBUTE</td>
<td>3-2</td>
</tr>
</tbody>
</table>
entry-namelist

ENVIRONMENT

ERROR

EVENT

EXCLUSIVE

EXIT

exit-statement

expression

expression-five

expression-four

expression-one

expression-six

expression-three

expression-two

EXTERNAL

external-option (implementation-defined)

extralingual-character (implementation-defined)

FILE

file-attribute

file-name-attribute

FINISH

FIXED

fixed-constant

FIXEDOVERFLOW

FLOAT

float-constant
format
FORMAT
format-item
format-iteration
format-sentence
formatlist
FREE
free-statement
FROM
function-attribute
G
GENERIC
generic-attribute
GET
GO
GOTO
goto-statement
group
H
I
IDENT
identifier
IF
if-clause
if-statement
IGNORE
imaginary-constant
IN
<table>
<thead>
<tr>
<th>INITIAL</th>
<th>3-3(26)</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial-attribute</td>
<td>3-2(15), 3-3(26), 3-9(89)</td>
</tr>
<tr>
<td>initial-call</td>
<td>3-3(26), 3-3(27)</td>
</tr>
<tr>
<td>initial-constant</td>
<td>3-3(29), 3-3(30), 3-3(31)</td>
</tr>
<tr>
<td>initial-item</td>
<td>3-3(26), 3-3(28), 3-3(30)</td>
</tr>
<tr>
<td>initial-itemlist</td>
<td>3-3(26), 3-3(28), 3-3(30)</td>
</tr>
<tr>
<td>initial-iteration</td>
<td>3-3(29), 3-3(30)</td>
</tr>
<tr>
<td>initial-label</td>
<td>3-7(64), 3-7(65)</td>
</tr>
<tr>
<td>INPUT</td>
<td>3-4(40)</td>
</tr>
<tr>
<td>INTERNAL</td>
<td>3-5(42)</td>
</tr>
<tr>
<td>INTO</td>
<td>3-12(113)</td>
</tr>
<tr>
<td>io-condition</td>
<td>3-10(97), 3-10(98)</td>
</tr>
<tr>
<td>io-option (implementation-defined)</td>
<td>3-4(39)</td>
</tr>
<tr>
<td>IRREDUCIBLE</td>
<td>3-9(34)</td>
</tr>
<tr>
<td>isub</td>
<td>3-13(123), 3-14(134)</td>
</tr>
<tr>
<td>iterated-group</td>
<td>3-7(69), 3-8(71)</td>
</tr>
<tr>
<td>J</td>
<td>3-14(131)</td>
</tr>
<tr>
<td>K</td>
<td>3-14(131), 3-15(148)</td>
</tr>
<tr>
<td>KEY</td>
<td>3-10(98), 3-12(113)</td>
</tr>
<tr>
<td>KEYED</td>
<td>3-4(40)</td>
</tr>
<tr>
<td>KEYFROM</td>
<td>3-12(113)</td>
</tr>
<tr>
<td>KEYTO</td>
<td>3-12(113)</td>
</tr>
<tr>
<td>L</td>
<td>3-14(131), 3-15(145)</td>
</tr>
<tr>
<td>LABEL</td>
<td>3-3(21)</td>
</tr>
<tr>
<td>label-attribute</td>
<td>3-2(15), 3-3(21)</td>
</tr>
<tr>
<td>labellist</td>
<td>3-1(7), 3-1(9), 3-5(44), 3-6(59), 3-7(64), 3-7(68), 3-8(75)</td>
</tr>
<tr>
<td>letter</td>
<td>3-14(130), 3-14(131), 3-14(132)</td>
</tr>
<tr>
<td>LIKE</td>
<td>3-5(43)</td>
</tr>
<tr>
<td>Key</td>
<td>Reference Numbers</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>like-attribute</td>
<td>3-2(14), 3-5(43)</td>
</tr>
<tr>
<td>LINE</td>
<td>3-6(54), 3-11(105)</td>
</tr>
<tr>
<td>LINESIZE</td>
<td>3-11(101)</td>
</tr>
<tr>
<td>LIST</td>
<td>3-11(109)</td>
</tr>
<tr>
<td>list-directed</td>
<td>3-11(106), 3-11(109)</td>
</tr>
<tr>
<td>LOCATE</td>
<td>3-12(112)</td>
</tr>
<tr>
<td>MULTIDIMENSION</td>
<td>3-14(131), 3-15(148)</td>
</tr>
<tr>
<td>N</td>
<td>3-14(131)</td>
</tr>
<tr>
<td>NAME</td>
<td>3-9(86), 3-10(98)</td>
</tr>
<tr>
<td>named-io-condition</td>
<td>3-10(94), 3-10(97)</td>
</tr>
<tr>
<td>no-check-condition</td>
<td>3-6(61), 3-10(96)</td>
</tr>
<tr>
<td>no-prefix</td>
<td>3-6(61), 3-7(63)</td>
</tr>
<tr>
<td>NOCHECK</td>
<td>3-10(96)</td>
</tr>
<tr>
<td>NOCONVERSION</td>
<td>3-7(63)</td>
</tr>
<tr>
<td>NOFIXEDOVERFLOW</td>
<td>3-7(63)</td>
</tr>
<tr>
<td>NOLOCK</td>
<td>3-12(113)</td>
</tr>
<tr>
<td>non-data-attribute</td>
<td>3-2(14), 3-4(33)</td>
</tr>
<tr>
<td>NOOVERFLOW</td>
<td>3-2(15), 3-7(63)</td>
</tr>
<tr>
<td>NORMAL</td>
<td>3-7(63)</td>
</tr>
<tr>
<td>NOSIZE</td>
<td>3-7(63)</td>
</tr>
<tr>
<td>NOSTRINGRANGE</td>
<td>3-7(63)</td>
</tr>
<tr>
<td>NOSUBSCRIPTRANGE</td>
<td>3-7(63)</td>
</tr>
<tr>
<td>NOUNDERFLOW</td>
<td>3-7(63)</td>
</tr>
<tr>
<td>NOZERODIVIDE</td>
<td>3-7(63)</td>
</tr>
<tr>
<td>null-statement</td>
<td>3-7(66), 3-7(67)</td>
</tr>
<tr>
<td>O</td>
<td>3-14(131)</td>
</tr>
<tr>
<td>OFFSET</td>
<td>3-3(22)</td>
</tr>
<tr>
<td>offset-attribute</td>
<td>3-2(15), 3-3(22), 3-4(37)</td>
</tr>
<tr>
<td>ON</td>
<td>3-10(91)</td>
</tr>
<tr>
<td>on-statement</td>
<td>3-7(66), 3-10(91)</td>
</tr>
</tbody>
</table>
proper-statement .............................................. 3-6 (59), 3-7 (66)

PUT ................................................................. 3-11 (104)

Q ................................................................. 3-14 (131)

R ................................................................. 3-15 (148)

READ ........................................................... 3-12 (112)

REAL ............................................................ 3-2 (16)

real-constant .................................................. 3-4 (32), 3-13 (128), 3-14 (136), 3-14 (139)

real-format ..................................................... 3-5 (49), 3-5 (50), 3-5 (51), 3-5 (51)

RECORD .......................................................... 3-4 (40), 3-10 (98)

record-io-statement ........................................... 3-7 (66), 3-12 (112)

record-optionslist ........................................... 3-12 (112), 3-12 (113)

RECURSIVE ..................................................... 3-1 (5)

REDUCIBLE ...................................................... 3-4 (34)

reference ........................................................ 3-3 (22), 3-3 (25), 3-6 (55), 3-8 (72), 3-8 (77), 3-8 (79), 3-9 (82), 3-9 (86), 3-9 (88), 3-9 (88), 3-9 (88), 3-9 (88), 3-10 (90), 3-10 (90), 3-11 (105), 3-12 (113), 3-12 (113), 3-12 (113), 3-12 (113), 3-12 (113), 3-12 (113), 3-12 (114), 3-12 (114), 3-12 (114), 3-12 (114), 3-13 (123), 3-13 (124), 3-13 (124)

template ....................................................... 3-5 (48), 3-6 (51)

replicated-string-constant ................................... 3-3 (31), 3-13 (128), 3-13 (129)

REPLY .......................................................... 3-12 (114), 3-12 (114)

RETURN .......................................................... 3-9 (81)

returns .......................................................... 3-7 (66), 3-9 (81)

REVERT .......................................................... 3-4 (34)

revert-statement .............................................. 3-7 (66), 3-10 (92)

rewite ........................................................... 3-12 (112)

S ................................................................. 3-14 (131), 3-15 (148)

scope-attribute ................................................ 3-2 (14), 3-5 (42)

SECONDARY ..................................................... 3-2 (15)

sentence ......................................................... 3-1 (6), 3-1 (8)

sentencelist ................................................... 3-1 (2), 3-1 (6), 3-7 (68), 3-7 (70), 3-8 (71)
SEQUENTIAL

SET

SECTS

SIGNAL

signal-statement

signed-integer

simple-group

simple-statement

simple-string-constant

SIZE

SKIP

SNAP

space

specification

statement

STATIC

sterling-constant

STOP

stop-statement

storage-class-attribute

STREAM

stream-io-statement

stream-optionslist

STRING

string-attribute

string-character

string-format

STRINGRANGE

SUB

SUBSCRIPRANGE
<table>
<thead>
<tr>
<th>Term</th>
<th>Usage Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM</td>
<td>3-10(91)</td>
</tr>
<tr>
<td>T</td>
<td>3-14(131), 3-15(148)</td>
</tr>
<tr>
<td>TASK</td>
<td>3-2(15), 3-8(79)</td>
</tr>
<tr>
<td>THEN</td>
<td>3-8(75)</td>
</tr>
<tr>
<td>TITLE</td>
<td>3-11(101)</td>
</tr>
<tr>
<td>TO</td>
<td>3-8(73), 3-8(73), 3-8(77)</td>
</tr>
<tr>
<td>TRANSMIT</td>
<td>3-10(98)</td>
</tr>
<tr>
<td>U</td>
<td>3-14(131)</td>
</tr>
<tr>
<td>UNBUFFERED</td>
<td>3-4(40)</td>
</tr>
<tr>
<td>unconditional-statement</td>
<td>3-6(57), 3-6(59), 3-8(76), 3-10(91)</td>
</tr>
<tr>
<td>UNDEFINEDFILE</td>
<td>3-10(98)</td>
</tr>
<tr>
<td>UNDERFLOW</td>
<td>3-6(62)</td>
</tr>
<tr>
<td>UNLOCK</td>
<td>3-12(112)</td>
</tr>
<tr>
<td>unqualified-reference</td>
<td>3-13(125), 3-13(126)</td>
</tr>
<tr>
<td>unsubscripted-reference</td>
<td>3-4(38), 3-5(43), 3-10(95), 3-10(96), 3-13(127)</td>
</tr>
<tr>
<td>UPDATE</td>
<td>3-4(40)</td>
</tr>
<tr>
<td>USES</td>
<td>3-4(34)</td>
</tr>
<tr>
<td>uses-sets-item</td>
<td>3-4(34), 3-4(38)</td>
</tr>
<tr>
<td>V</td>
<td>3-14(131), 3-15(148)</td>
</tr>
<tr>
<td>VARYING</td>
<td>3-2(15), 3-4(37)</td>
</tr>
<tr>
<td>W</td>
<td>3-14(131)</td>
</tr>
<tr>
<td>WAIT</td>
<td>3-9(82)</td>
</tr>
<tr>
<td>wait-statement</td>
<td>3-7(66), 3-9(82)</td>
</tr>
<tr>
<td>WHILE</td>
<td>3-8(71), 3-8(73)</td>
</tr>
<tr>
<td>WRITE</td>
<td>3-12(112)</td>
</tr>
<tr>
<td>X</td>
<td>3-6(54), 3-14(131), 3-15(148)</td>
</tr>
<tr>
<td>Y</td>
<td>3-14(131), 3-15(148)</td>
</tr>
<tr>
<td>Z</td>
<td>3-14(131), 3-15(148)</td>
</tr>
<tr>
<td>ZERODIVIDE</td>
<td>3-6(62)</td>
</tr>
<tr>
<td>0</td>
<td>3-14(133), 3-15(142)</td>
</tr>
<tr>
<td></td>
<td>3-14 (133), 3-15 (142), 3-15 (148)</td>
</tr>
<tr>
<td>---</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3-14 (133), 3-15 (148)</td>
</tr>
<tr>
<td>3</td>
<td>3-14 (133), 3-15 (148)</td>
</tr>
<tr>
<td>4</td>
<td>3-14 (133)</td>
</tr>
<tr>
<td>5</td>
<td>3-14 (133)</td>
</tr>
<tr>
<td>6</td>
<td>3-14 (133), 3-15 (148)</td>
</tr>
<tr>
<td>7</td>
<td>3-14 (133), 3-15 (148)</td>
</tr>
<tr>
<td>8</td>
<td>3-14 (133), 3-15 (148)</td>
</tr>
<tr>
<td>9</td>
<td>3-14 (133), 3-15 (148)</td>
</tr>
</tbody>
</table>