A
ANS COBOL reserved words

ACCEPT	CF	DATE
ACCESS	CH	DATE-COMPILED
ADD	CHARACTER	DATE-WRITTEN
ADVANCING	CHARACTERS	DAY
AFTER	CLOCK-UNITS	DE
ALL	CLOSE	DEBIJG-CONTENTS
ALPHABETIC	COBOL	DEBUG-ITEM
ALSO	CODE	DEBUG-LINE
ALTER	CODE-SET	DEBUG-NAME
ALTERNATE	COLLATING	DEBUG-SUB-I
AND	COLUMN	DEBUG-SUB-2
ARE	COMMA	DEBUG-SUB-3
AREA	COMMUNICATION	DEBUGGING
AREAS	COMP	DECIMAL-POINT
ASCENDING	COMPUTATIONAL	DECLARATIVES
ASSIGN	COMPUTE	DELETE
AT	CONFIGURATION	DELIMITED
AUTHOR	CONTAINS	DELIMITER
BEFORE	CONTROL	DEPENDING
BLANK	CONTROLS	DESCENDING
BLOCK	COPY	DESTINATION
BOTTOM	CORK	DETAIL
BY	CORRESPONDING	DISABLE
CALL	COUNT	DISPLAY
CANCEL	CURRENCY	DIVIDE
CD	DATA	DIVISION

DOWN
DUPLICATES
DYNAMIC
EGI
ELSE
EM1
ENARLE
END
END-OF-PAGE
ENTER
ENVIRONMENT
EOP
EQUAL
ERROR ESI
EVERY
EXCEPTION
EXIT
EXTEND
FD
FILE
FILE-CONTROL
FILLER
FINAL
FIRST
FOOTING
FOR
FROM
GENERATE
GIVING
GO
GREATER
GROUP
HEADING
HIGH-VALUE
HIGH-VALUES
I-O
I-O-CONTROL
IDENTIFICATION
IF
IN
INDEX
INDEXED

INDICATE
INITIAL
INITIATE
INPUT
INPUT-OUTPUT
INSPECT
INSTALLATION
INTO
INVALID
IS
JUST
JUSTIFIED
KEY
LABEL
LAST
LEADING
LEFT
LENGTH
LESS
LIMIT
LIMITS
LINAGE
LINAGE-COUNTER
LINE
LINE-COUNTER
LINES
LINKAGE
LOCK
LOW-VALUE
LOW-VALUES
MEMORY
MERGE
MESSAGE
MODE
MODULES
MOVE
MULTIPLE
MULTIPLY
NATIVE
NEGATIVE
NEXT
NO

NOT
NUMBER
NUMERIC
OBJECT-COMPUTER
OCCURS
OF
OFF
OMITTED
ON
OPEN
OPTIONAL
OR
ORGANIZATION
OUTPUT
OVERFLOW
PAGE
PAGE-COUNTER
PERFORM
PF
PH
PIC
PICTURE
PLUS
POINTER
POSITION
POSITIVE
PRINTING
PROCEDURE
PROCEDURES
PROCEED
PROGRAM
PROGRAM-ID
QUEUE
QUOTE
QUOTES
RANDOM
RD
READ
RECEIVE
RECORD
RECORDS
REDEFINES

REEL
REFERENCES
RELATIVE
RELEASE
REMAINDER
REMOVAL
RENAMES
REPLACING
REPORT
REPORTING
REPORTS
RERUN
RESERVE
RESET
RETURN
REVERSED
REWIND
REWRITE
RF
RH
RIGHT
ROUNDED
RUN
SAME
SD
SEARCH
SECTION
SECURITY
SEGMENT
SEGMENT-LIMIT
SELECT
SEND
SENTENCE
SEPARATE
SEQUENCE
SEQUENTIAL
SET
SIGN
SIZE
SORT
SORT-MERGO
SOURCE
SOURCE-COMPUTER
SPACE
SPACES
SPECIAL-NAMES
STANDARD
STANDARD-I
START
STATUS
STOP
STRING
SUB-QUEUE-t
SUB-QUEUE-2
SUB-QUEUE-3
SUBTRACT
SUM
SUPPRESS
SYMBOLIC
SYNC
SYNCHRONIZED
TABLE
TALLYING
TAPE
TERMINAL
TERMINATE
TEXT
THAN
THROUGH
THRU
TIME
TIMES
TO
TOP
TRAILING
TYPE
UNIT
UNSTRING
UNTIL
UP
UPON
USAGE
USE
USING
VALUE
VALUES
VARYING
WHEN
WITH
WORDS
WORKING-STORAGE
WRITE
ZERO
ZEROES
ZEROS
+
-
*
/
**
>
<
=

B
Complete ANS COBOL language formats

The following set of conventions is followed in the format statements:

1 Words presented in uppercase are always reserved COBOL words

2 Uppercase words that are underlined are words that are required in the type of program statement being described. Uppercase words that are not underlined are optional and are used only to improve the readability of the program.

3 Lowercase words are used to indicate the points at which data-names or constants are to be supplied by the programmer. In addition to the words "data-name" and "literal," the term "identifier" is used to indicate a data-name, but it has a slightly broader meaning. It refers to either of the following cases: data-names that are unique in themselves, or data-names that are not unique in themselves but are made unique through qualification. Qualification is discussed in Chapter 8. Other lowercase words used to indicate items to be ins&ted by the programmer are:

   file-name
   record-name
   integer
   formula
   condition
   statement
   any imperative statement
   any sentence

4 Items enclosed in braces{) indicate that one of the enclosed items must be used.

5 Items enclosed in brackets[ ] indicate that the items are optional, and one of them may be used, at the discretion of the programmer.

6 An ellipsis ( ) indicates that further information maybe included in the program instruction, usually in the form of repeating the immediately preceding element any desired number of times.

GENERAL FORMAT FOR IDENTIFICATION DIVISION

```
IDENTIFICATION DIVISION.
PROGRAM-ID. program-name.
  [AUTHOR. [comment-entry] ,
  [INSTALLATION. [comment-entry] . ,
  [DATE-WRITTEN. [comment-entry] . ,
  [DATE-COMPILED. [comment-entry] . ,
  [SECURITY. [comment-entry] .. ]
```

GENERAL FORMAT FOR ENVIRONMENT DIVISION

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. computer-name [WITH DEBUGGING MODE].
OBJECT-COMPUTER. computer-name
  [                          { WORDS      } ]
  [ MEMORY SIZE integer      { CHARACTERS } ]
  [                          { MODULES    } ]
  [, PROGRAM COLLATING SEQUENCE IS alphabet-name]
  [, SEGMENT-LIMIT IS segment-number].
[SPECIAL-NAMES. [, implementor-name
  { IS mnemonic name [, ON STATUS IS condition-name-1                          } ]
  {     [, OFF STATUS IS condition-name-2]]                                    } ]
  { IS mnemonic-name [, OFF STATUS IS condition-name-2                         } ]
  {     [, ON STATUS IS condition-name-1]]                                     } ]
  { ON STATUS IS condition-name-1 [, OFF STATUS IS condition-name-2]           } ]
  { OFF STATUS IS condition-name-2 [, ON STATUS IS condition-name-1]           } ]
```

```
[                     ( STANDARD-1                                                    )]
[                     ( NATIVE                                                        )]
[ , alphabet-name IS  ( implementor-name                                              )]
[                     (            [ { THROUGH }  literal-2                     ]     )]
[                     ( literal-1  [ { THRU    }                                ]     )]
[                     (            [ ALSO literal-3 [, ALSO literal-4] . . .    ]     )] . . .
[                     (    [            [ { THROUGH }  literal-6          ] ]         )]
[                     (    [ literal-5  [ { THRU    }                     ] ] . . .   )]
[                     (    [            [ ALSO literal-7 [, ALSO literal-8] . . . ] ] )]
 [, CURRENCY SIGN IS literal-9]
 [, DECIMAL-POINT IS COMMA]
[INPUT-OUTPUT SECTION.
FILE-CONTROL.
   {file-control-entry}

[I-O-CONTROL.
 [ ; RERUN [ON { file-name-1      } ]
           [   { implementor-name } ]
          ( { [END OF] { REEL } }  OF file-name-2 )
          ( {          { UNIT } }                 )
  EVERY   ( { integer-1 RECORDS }                 ) . . .
          ( integer-2 CLOCK-UNITS                 )
          ( condition-name                        )
 [          [ RECORD     ]                                            ]
 [ ; SAME   [ SORT       ]  AREA FOR file-name-3 {, file-name-4} . . .]
 [          [ SORT-MERGE ]                                            ]
 [; MULTIPLE FILE TAPE CONTAINS file-name-5 [POSITION integer-3]
      [, file-name-6 [POSITION integer-4]] . . . ] . . . .]]
```

GENERAL FORMAT FOR FILE CONTROL ENTRY

FORMAT 1

```
SELECT [OPTIONAL] file-name
  ASSIGN TO implementor-name-1 [, implementor-name-2] . . .
  [ ; RESERVE integer-1 [ AREA  ] ]
  [                     [ AREAS ] ]
  [; ORGANIZATION IS SEQUENTIAL]
```

```
    , ; ACCESS MODE IS SEQUENTIAL,
    , ; FILE STATUS IS data-name-1]
```

FORMAT 2

```
SELECT file-name
  ASSIGN TO implementor-name-1 [, implementor-name-2] . . .
   [ ; RESERVE integer-1 [ AREA  ] ]
                         [ AREAS ]
     ; ORGANIZATION IS RELATIVE
   [                    { SEQUENTIAL [, RE    TIVE KEY IS data-name-1] }
   [ ; ACCESS MODE IS   {                                              }
   [                    { { RANDOM  }  RELATIVE KEY IS data-name-1     }
                        { { DYNAMIC }                                  }
   [ ; FILE STATUS IS data-name-2] .
```

FORMAT 3

```
SELECT file-name
  ASSIGN TO implementor-name-1 [, implementor-name-2] . .
   [ ; RESERVE integer-1 [ AREA  ] ]
                         [ AREAS ]
     ; ORGANIZATION IS INDEXED
   [                    { SEQUENTIAL } ]
   [ ; ACCESS MODE IS   { RANDOM     } ]
   [                    { DYNAMIC    } ]
     ; RECORD KEY IS data-name-1
   [ ; ALTERNATE RECORD KEY IS data-name-2 [WITH DUPLICATES, ,
    , ; FILE STATUS IS data-name-3].
```

FORMAT 4

```
SELECT file-name ASSIGN TO implementor-name-1 [, implementor-name-2]
```

GENERAL FORMAT FOR DATA DIVISION

```
DATA DIVISION.

[FILE SECTION.

[FD  file-name

    [ ; BLOCK CONTAINS  [integer-1 TO]  integer-2 { RECORDS    } ]
                                                  { CHARACTERS }

    [ ; RECORD CONTAINS  [integer-3 TO]  integer-4 CHARACTERS]

      ; LABEL { RECORD IS    }  { STANDARD }
              { RECORDS ARE  }  { OMITTED  }

    [ ; VALUE OF implementor-name-1 IS { data-name-1 }
                                       { literal-1   }

          [ , implementor-name-2 IS { data-name-2 } ]  . . . ]
                                    { literal-2   }

    [ ; DATA { RECORD IS    } data-name-3  [, data-name-4] . . . ]
             { RECORDS ARE  }

    [ ; LINAGE IS { data-name-5 } LINES [ , WITH FOOTING AT { data-name-6 } ]
                  { integer-5   }                           { integer-6   }

         [ , LINES AT TOP { data-name-7 } ]  [ , LINES AT BOTTOM { data-name-8 } ] ]
                          { integer-7   }                        { integer-8   }

   [; CODE-SET IS alphabet-name]

    [ ; { REPORT IS    } report-name-1  [, report-name-2] . . . ] .
        { REPORTS ARE  }

[record-description-entry] . . . ] . . .

[SD  file-name

    [ ; RECORD CONTAINS [integer-1 TO] integer-2 CHARACTERS]

    [ ; DATA { RECORD IS    } data-name-1  [, data-name-2] . . . ]
             { RECORDS ARE  }

{record-description-entry} . . . ] . . .

[WORKING-STORAGE SECTION.

[ 77-level-description-entry ]  . . . ]
[ record-description-entry   ]

[LINKAGE SECTION.

[ 77-level-description-entry ]  . . . ]
[ record-description-entry   ]
```

```
[COMMUNICATION SECTION.
[communication-description-entry
[record-description-entry] . . . ] . . . ]

[REPORT SECTION.
[RD report-name
[; CODE literal-1]
[ ; { CONTROL IS   } { data-name-1 [, data-name-2] . . .                  } ]
[   { CONTROLS ARE } { FINAL [, data-name-1 [, data-name-2] . . . ]      } ]
[ ; PAGE [ LIMIT IS   ] integer-1 [ LINE  ] [, HEADING integer-2]
[        [ LIMITS ARE ]           [ LINES ]
   [, FIRST DETAIL integer-3] [, LAST DETAIL integer-4]
   [, FOOTING integer-5]].
{report-group-description-entry}    ]    ]
```

GENERAL FORMAT FOR DATA DESCRIPTION ENTRY

FORMAT 1

```
level-number { data-name-1 }
             { FILLER      }
  [; REDEFINES data-name-2]
  [ ; { PICTURE } I S character-string ]
  [   { PIC     }                      ]
  [                  { COMPUTATIONAL } ]
  [ ; [USAGE IS]     { COMP          } ]
  [                  { DISPLAY       } ]
  [                  { INDEX         } ]
  [ ; [SIGN IS] { LEADING  } [SEPARATE CHARACTER] ]
  [             { TRAILING }                      ]
  [ ; OCCURS { integer-1 TO integer-2 TIMES DEPENDING ON data-name-3 }
  [          { integer-2 TIMES                                       }
     [ { ASCENDING  } KEY IS data-name-4 [, data-name-5] . . . ] . . .
     [ { DESCENDING }                                          ]
     [INDEXED BY index-name-1 [, index-name-2] . . . ] ]
  [ ; { SYNCHRONIZED } [ LEFT  ] ]
  [   { SYNC         } [ RIGHT ] ]
[ ; { JUSTIFIED } RIGHT ]
    { JUST      }
[; BLANK WHEN ZERO]
[; VALUE IS literal]
```

FORMAT 2

```
66 data-name-1; RENAMES data-name-2 [ { THROUGH } data-name-3 ]
                                      { THRU    }
```

FORMAT 3

```
88  condition-name; { VALUE IS   } literal-1 [ { THROUGH } literal-2 ]
                    { VALUES ARE }             { THRU    }
    [, literal-3 [ { THROUGH } literal-4 ] ] . . .
                   { THRU    }
```

GENERAL FORMAT FOR COMMUNICATION DESCRIPTION ENTRY

FORMAT 1

```
CD  cd-name:
                           [ [; SYMBOLIC QUEUE IS data-name-1]
                               [; SYMBOLIC SUB-QUEUE-1 IS data-name-2]
                               [; SYMBOLIC SUB-QUEUE-2 IS data-name-3]
                               [; SYMBOLIC SUB-QUEUE-3 IS data-name-4]
                               [; MESSAGE DATE IS data-name-5]
                               [; MESSAGE TIME IS data-name-6]
FOR [INITIAL] INPUT            [; SYMBOLIC SOURCE IS data-name-7]
                               [; TEXT LENGTH IS data-name-8]
                               [; END KEY IS data-name-9]
                               [; STATUS KEY IS data-name-10]
                               [; MESSAGE COUNT IS data-name-11]
                           [data-name-1, data-name-2, . data-name-11] ]
```

FORMAT 2

```
CD cd-name; FOR OUTPUT
  [; DESTINATION COUNT IS data-name-1]
  [; TEXT LENGTH IS data-name-2]
  [; STATUS KEY IS data-name-3]
  [; DESTINATION TABLE OCCURS integer-2 TIMES
       [; INDEXED BY index-name-1 [, index-name-2]...]]
  [; ERROR KEY IS data-name-4]
  [; SYMBOLIC DESTINATION IS data-name-5].
```

GENERAL FORMAT FOR REPORT GROUP DESCRIPTION ENTRY

FORMAT 1

```
01  [data-name-1]

    [ ; LINE NUMBER IS { integer-1 [ON NEXT PAGE] } ]
    [                  { PLUS integer-2           } ]

    [ ; NEXT GROUP IS { integer-3      } ]
    [                 { PLUS integer-4 } ]
    [                 { NEXT PAGE      } ]

                 ( { REPORT HEADING }                                  )
                 ( { RH             }                                  )
                 ( { PAGE HEADING }                                    )
                 ( { PH           }                                    )
                 ( { CONTROL HEADING }  { data-name-2 }                )
                 ( { CH              }  { FINAL       }                )
    ; TYPE IS    ( { DETAIL }                                          )
                 ( { DE     }                                          )
                 ( { CONTROL FOOTING }  { data-name-3 }                )
                 ( { CF              }  { FINAL       }                )
                 ( { PAGE FOOTING }                                    )
                 ( { PF           }                                    )
                 ( { REPORT FOOTING }                                  )
                 ( { RF             }                                  )

    [; [USAGE IS] DISPLAY ]
```

FORMAT 2

```
level-number [data-name-1]

    [; LINE NUMBER IS { integer-1 [ON NEXT PAGE] } ]
                      { PLUS integer-2           }
    [; [USAGE IS, DISPLAY ]
```

FORMAT 3

```
level-number  [data-name-1]

    [ ; BLANK WHEN ZERO]

    [ ; GROUP INDICATE]

    [ ; { JUSTIFIED } RIGHT ]
        { JUST      }

    [ ; LINE NUMBER IS { integer-1 [ON NEXT PAGE] } ]
                       { PLUS integer-2           }

    [ ; COLUMN NUMBER IS integer-3]

      ; { PICTURE } IS character-string
        { PIC     }

   { ; SOURCE IS identifier-1                              }
   { ; VALUE IS literal                                    }
   {{ ; SUM identifier-2 [, identifier-3] . .              }
   {      [UPON  data-name-2  [, data-name-3] . .  ]}.     }
   {      [ RESET ON { data-name-4 } ]                     }
   {                 { FINAL       }                       }

    , ; [USAGE IS, DISPLAY].
```

GENERAL FORMAT FOR PROCEDURE DIVISION

FORMAT 1

```
PROCEDURE DIVISION [USING data-name-1 [, data-name-2] . ]
[DECLARATIVES.
{section-name SECTION [segment-number]. declarative-sentence
[paragraph-name. [sentence]     ]..   }
END DECLARATIVES.]
{section-name SECTION [segment-number].
[paragraph-name. [sentence]     ].   }
```

FORMAT 2

```
PROCEDURE DIVISION [USING data-name-1 [, data-name-2]. .]
{paragraph-name. [sentence].        }
```

GENERAL FORMAT FOR VERBS

```
ACCEPT identifier [FROM mnemonic-name]

                         { DATE }
ACCEPT identifier FROM   { DAY  }
                         { TIME }

ACCEPT cd-name MESSAGE COUNT

    { identifier-1 }  [ , identifier-2 ]
ADD { literal-1    }  [ , literal-2    ] . . . TO identifier-m [ROUNDED]
    [, identifier-n [ROUNDED]]        [; ON SEE ERROR imperative-statement]

    { identifier-1 }    { identifier-2 }  [ , identifier-3 ]
ADD { literal-1    }  , { literal-2    }  [ , literal-3    ]
    GIVING identifier-m [ROUNDED] [, identifier-n [ROUNDED]].
    [; ON SIZE ERROR imperative-statement]

    { CORRESPONDING }
ADD { CORR          }  identifier 1 TO ientifier-2 [ROUNDED]
    [; ON SIZE ERROR imperative-statement]

ALTER procedure-name-1 TO [PROCEED TO] procedure-name-2
    [, procedure-name-3 TO [PROCEED TO, procedure-name-4].

     { identifier-1 }
CALL { literal-1    }  [USING data-name-1 [, data-name-2] . . . ]
```

```
    [; ON OVERFLOW imperative-statement]

CANCEL { identifier-1 }  [ , identifier-2 ] ...
       { ~~literal-1~~ }  [ , literal-2    ]

                       [ { REEL }  [ WITH NO REWIND ] ]
                       [ { UNIT }  [ FOR REMOVAL    ] ]
CLOSE file-name-1      [                              ]
                       [ WITH { NO REWIND }           ]
                       [      { LOCK      }           ]

    [                 [ { REEL }  [ WITH NO REWIND ] ] ]
    [                 [ { UNIT }  [ FOR REMOVAL    ] ] ]
    [ , file-name-2   [                              ] ] ...
    [                 [ WITH { NO REWIND }           ] ]
    [                 [      { LOCK      }           ] ]

CLOSE file-name-1 [WITH LOCK] [, file-name-2 [WITH LOCK]] ...

COMPUTE identifier-1 [ROUNDED] [, identifier-2 [ROUNDED]] ...
    = arithmetic-expression [; ON SIZE ERROR imperative-statement]

DELETE file-name RECORD [; INVALID KEY imperative-statement]

DISABLE { INPUT [TERMINAL] }  cd-name WITH KEY { identifier-1 }
        { OUTPUT           }                   { literal-1    }

DISPLAY { identifier-1 }  { , identifier-2 } ... [UPON mnemonic-name]
        { literal-1    }  { , literal-2    }

DIVIDE { identifier-1 }  INTO identifier-2 [ROUNDED]
       { literal-1    }

    [, identifier-3 [ROUNDED]] ... [; ON SIZE ERROR imperative-statement]

DIVIDE { identifier-1 }  INTO { identifier-2 }  GIVING identifier-3 [ROUNDED]
       { literal-1    }       { literal-2    }

    [, identifier-4 [ROUNDED]] ... [; ON SIZE ERROR imperative-statement]

DIVIDE { identifier-1 }  BY { identifier-2 }  GIVING identifier-3 [ROUNDED]
       { literal-1    }     { literal-2    }

    [, identifier-4 [ROUNDED]] ... [; ON SIZE ERROR imperative-statement]

DIVIDE { identifier-1 }  INTO { identifier-2 }  GIVING identifier-3 [ROUNDED]
       { literal-1    }       { literal-2    }

    REMAINDER identifier-4 [; ON SIZE ERROR imperative-statement]
```

```
DIVIDE  { identifier-1 }  BY  { identifier-2 }  GIVING identifier-3 IROUNDED]
        { literal-1    }      { literal-2    }

    REMAINDER identifier-4 [; ON SIZE ERROR imperative-statement]

ENABLE { INPUT [TERMINAL] }  cd-name WITH KEY { identifier-1 }
       { OUTPUT           }                   { literal-1    }

ENTER language-name [routine-name].

EXIT [PROGRAM,.

GENERATE { data-name   }
         { report-name }

GO TO [procedure-name-1 1

GO TO  procedure-name-1 [, procedure-name-2] ,  procedure "MC-"

    DEPENDING ON identifier

IF  condition; { statement-1   }  { ; ELSE statement-2          }
               { N-SENTENCE    }  { ; ELSE NEXT SENTENCE         }

INITIATE report-name-1 [, report-name-2].

INSPECT identifier-1 TALLYING

    {                         {  { ALL     }   { identifier-3 } }
    { , identifier-2 FOR      {  { LEADING }   { literal-1    } }
    {                         {  CHARACTERS                     }

          [ { BEFORE }  INITIAL { identifier-4 } ] ...  } ...  }
          [ { AFTER  }          { literal-2    } ]

INSPECT identifier-1 REPLACING

    { CHARACTERS BY { identifier-6 }    [ { BEFORE }  INITIAL { identifier-7 } ]
    {               { literal-4    }    [ { AFTER  }          { literal-5    } ]

      {   { ALL     }  {  { identifier-5 }  BY  { identifier-6 }
      { , { LEADING }  {  { literal-3    }      { literal-4    }
      {   { FIRST   }  {

          [ { BEFORE }  INITIAL { identifier-7 } ] }  ...  }  ...  }
          [ { AFTER  }          { literal-5    } ] }

INSPECT identifier-1 TALLYING

    {                         {  { ALL     }   { identifier-3 } }
    { , identifier-2 FOR      {  { LEADING }   { literal-1    } }
    {                         {  CHARACTERS                     }
```

```
        [{ BEFORE } INITIAL { identifier-4 }]  } ...  { ...
        [{ AFTER  }         { literal-2    }]  }      {

     REPLACING

     { CHARACTERS BY { identifier-6 }    { BEFORE } INITIAL { identifier-7 }]
     {               { literal-4    }    { AFTER  }         { literal-5    }]

      {  { ALL     }  {   { identifier-5 }  BY  { identifier-6 }
      {  { LEADING }  {   { literal-3    }      { literal-4    }
      {  { FIRST   }  {

           [{ BEFORE } INITIAL { identifier-7 }]  } ...  { ...  }
           [{ AFTER  }         { literal-5    }]  }      {      }

   MERGE file-name-1   ON { ASCENDING  } KEY data-name-1 [, data-name-2] ...
                          { DESCENDING }

                  [ ON { ASCENDING  } KEY data-name-3 [, data-name-4] ... ] ..
                  [    { DESCENDING }                                     ]

     [COLLATING SEQUENCE IS alphabet-name]

     USING file-name-2, file-name-3 [, file-name-4] ...

     { OUTPUT PROCEDURE IS section-name-1  [{ THROUGH } section-name-2] }
     {                                     [{ THRU    }               ] }
     { GIVING file-name-5

   MOVE { identifier-1 }  TO identifier-2 [, identifier-3] ...
        { literal      }

   MOVE { CORRESPONDING } identifier-1 TO identifier-2
        { CORR          }

   MULTIPLY { identifier-1 }  BY identifier-2 [ROUNDED]
            { literal-1    }

        [, identifier-3 [ROUNDED]]  ... [; ON SIZE ERROR imperative-statement]

   MULTIPLY { identifier-1 }  BY { identifier-2 }  GIVING identifier-3 [ROUNDED]
            { literal-1    }     { literal-2    }

        [, identifier-4 [ROUNDED]]  [; ON  SIZE ERROR imperative-statement]

   OPEN  {  INPUT file-name-1  [ REVERSED       ]
         {                     [ WITH NO REWIND ]

                  [, file-name-2 [ REVERSED       ]]  ...
                  [              [ WITH NO REWIND ]]
```

```
          OUTPUT file-name-3  [WITH NO REWIND]
               [, file-name-4  [WITH  NO  REWIND]]
          I-O  file-name-5  [,  file-name-6] .
          EXTEND  file-name-7  [,  file-name-8]

        { INPUT file-name-1 [, file-name-2] . . .  }
OPEN    { OUTPUT file-name-3 [, file-name-4] . . . } . . .
        { I-O file-name-5 [, file-name-6] . . .    }

PERFORM procedure-name-1  [ { THROUGH }  procedure-name-2 ]
                          [ { THRU    }                   ]

PERFORM procedure-name-1  [ { THROUGH }  procedure-name-2 ]  { identifier-1 }  TIMES
                          [ { THRU    }                   ]  { integer-1    }

PERFORM procedure-name-1  [ { THROUGH }  procedure-name-2 ]  UNTIL condition-1
                          [ { THRU    }                   ]

PERFORM procedure-name-1  [ { THROUGH }  procedure-name-2 ]
                          [ { THRU    }                   ]

    VARYING { identifier-2 }  FROM { identifier-3 }
            { index-name-1 }       { index-name-2 }
                                   { literal-1    }

        BY { identifier-4 }  UNTIL condition-1
           { literal-3    }

   [  AFTER { identifier-5 }  FROM { identifier-6 }
   [        { index-name-3 }       { index-name-4 }
                                   { literal-3    }

        BY { identifier-7 }  UNTIL condition-2
           { literal-4    }

   [  AFTER { identifier-8 }  FROM { identifier-9 }
   [        { index-name-5 }       { index-name-6 }
                                   { literal-5    }

        BY { identifier-10 }  UNTIL condition-3 ]]
           { literal-6     }

READ file-name RECORD [INTO identifier] [; AT END imperative-statement]

READ file-name [NEXT] RECORD [INTO identifier]
    [; AT END imperative-statement]

READ file-name RECORD [INTO identifier] [; INVALID KEY imperative-statement]
READ file-name RECORD [INTO identifier]
    [; KEY IS data-name]
    [; INVALID KEY imperative statement]

RECEIVE cd-name { MESSAGE } INTO identifier-1 [; NO DATA imperative-statement]
                { SEGMENT }

RELEASE record-name [FROM identifier]

RETURN file-name RECORD [INTO identifier] ; AT END imperative-statement

REWRITE record-name [FROM identifier]

REWRITE record-name [FROM identifier] [; INVALID KEY imperative-statement]

SEARCH identifier-1 [ VARYING { identifier-2 } ]  [; AT END imperative-statement-1]
                    [         { index-name-1 } ]

      ; WHEN condition-1 { imperative-statement-2 }
                         { NEXT SENTENCE          }

    [ ; WHEN condition-2 { imperative-statement-3 } ] ...
    [                    { NEXT SENTENCE          } ]

SEARCH ALL identifier-1 [; AT END imperative-statement-1]

                 { data-name-1 { IS EQUAL TO } { identifier-3            } }
    ; WHEN       {             { IS =        } { literal-1               } }
                 {                             { arithmetic-expression-1 } }
                 { condition-name-1                                        }

          [      { data-name-2 { IS EQUAL TO } { identifier-4            } } ]
          [ AND  {             { IS =        } { literal-2               } } ] ...
          [      {                             { arithmetic-expression-2 } } ]
          [      { condition-name-2                                        } ]

          { imperative-statement-2 }
          { NEXT SENTENCE          }

SEND cd-name FROM identifier-1

                                   { WITH identifier-2 }
SEND cd-name [FROM identifier-1]   { WITH ESI          }
                                   { WITH EMI          }
                                   { WITH EGI          }

      [                        { { { identifier-3 }  [ LINE  ] } } ]
      [ { BEFORE } ADVANCING   { { { integer      }  [ LINES ] } } ]
      [ { AFTER  }             {                                 } ]
      [                        { { mnemonic-name }               } ]
      [                        { { PAGE          }               } ]
```

```
SET { identifier-1 [, identifier-2] . . .    }  TO  { identifier-3   }
    { index-name-1 [, index-name-2] . . .    }      { index-name-3   }
                                                    { integer-1      }

SET index-name-4 [, index-name-5] . . . { UP BY   }  { identifier-4 }
                                        { DOWN BY }  { integer-2    }

SORT file-name-1 ON { ASCENDING  } KEY data-name-1 [, data-name-2] . .
                    { DESCENDING }

             [ON { ASCENDING  } KEY data-name-3 [, data-name-4] . . ]
             [   { DESCENDING }                                     ]

    [COLLATING SEQUENCE IS alphabet-name]

    { INPUT PROCEDURE IS section-name-1   [ { THROUGH } section-name-2 ] }
    {                                     [ { THRU    }                ] }
    { USING file-name-2 [, file-name-3] .                                }

    { OUTPUT PROCEDURE IS section-name-3  [ { THROUGH } section-name-4 ] }
    {                                     [ { THRU    }                ] }
    { GIVING file-name-4                                                 }

                  [     { IS EQUAL TO         }           ]
                  [     { IS =                }           ]
START file-name   [ KEY { IS GREATER THAN     } data-name ]
                  [     { IS >                }           ]
                  [     { IS NOT LESS THAN    }           ]
                  [     { IS NOT <            }           ]

    [; INVALID KEY imperative-statement]

STOP { RUN     }
     { literal }

STRING { identifier-1 } [, identifier-2] . . . DELIMITED BY { identifier-3 }
       { literal-1    } [, literal-2   ]                    { literal-3    }
                                                            { SIZE         }

    [ { identifier-4 } [, identifier-5] . . . DELIMITED BY { identifier-6 } ] . .
    [ { literal-4    } [, literal-5   ]                    { literal-6    } ]
    [                                                      { SIZE         } ]

    INTO identifier-7 [WITH POINTER identifier-8]

    [; ON OVERFLOW imperative-statement]

SUBTRACT { identifier-1 } [, identifier-2] . . . FROM identifier-m [ROUNDED]
         { literal-1    } [, literal-2   ]

    [, identifier-n [ROUNDED]] . . . [; ON SIZE ERROR imperative-statement]
```

```
SUBTRACT { identifier-l }  [, identifier-2 ]  . . .  FROM* { identifier-m }
         { literal-1    }  [, literal-2    ]               { literal-m    }

    GIVING identifier-n [ROUNDED] [, identifier-o [ROUNDED] ]

    [; ON SIZE ERROR imperative-statement]

SUBTRACT { CORRESPONDING } identifier- l FROM identifier-2 [ROUNDED]
         { CORR          }

    [; ON SIZE ERROR imperative-statement]

SUPPRESS PRINTING

TERMINATE report-name-1 [, report-name-2]

UNSTRING identifier-1

   [ DELIMITED BY [ALL] { identifier-2 }  [, OR [ALL] { identifier-3 } ] . . . ]
                        { literal-1    }               { literal-2    }

    INTO identifier-4 [, DELIMITER IN identifier-5] [, COUNT IN identifier-6]

     [, identifier-7 [, DELIMITER IN identifier-8] [, COUNT IN identifier91 ]

    [WITH POINTER identifier-10] [TALLYING IN identifier-1 l]

    [; ON OVERFLOW imperative-statement]

                                                        ( file-name-
                                                        { INPUT
USE AFTER STANDARD { EXCEPTION } PROCEDUREON            { OUTPUT
                   { ERROR     }                        { I-O
                                                        ( EXTEND

                                                      ( file-name-l [, file-name-2] . )
USE AFTER STANDARD { EXCEPTION } PROCEDURE ON         { INPUT                         }
                   { ERROR     }                      { OUTPUT                        }
                                                      ( I-O                           )

USE BEFORE REPORTING identifier.

                        cd-name-l
                        [ALL REFERENCES OF, identifier- l
USE FOR DEBUGGING ON    file-"me-l
                        procedure-name-1
                      l ALL PROCEDURES

     [ cd-name-2
       [ALL REFERENCES OF, identifier-2
       file-name-2
       procedure-name-2
       ALL PROCEDURES                      ]
```

```
WRITE record-name [FROM identifier-1]

  [ { BEFORE }  ADVANCING  { { identifier-2 }  [ LINE  ] }  ]
  [ { AFTER  }             { { integer      }  [ LINES ] }  ]
  [                        { { mnemonic-name}            }  ]
  [                        { { PAGE         }            }  ]

  [; AT { END-OF-PAGE }  imperative-statement ]
  [     { EOP         }                       ]

WRITE record-name [FROM identifier1 [. INVALID KEY imperative-statement]
```

GENERAL FORMAT FOR CONDITIONS

RELATION CONDITION

```
                              { IS [NOT] GREATER THAN }
{ identifier-1              } { IS [NOT] LESS THAN    } { idsnrifk.2              }
{ literal-1                 } { IS [NOT] EQUAL TO     } { literal-2               }
{ arithmetic-expression-1   } { IS [NOT] >            } { arithmetic-expression-2 }
{ index-name-1              } { IS [NOT] <            } { index-dam-2             }
                              { IS [NOT] =            }
```

CLASS CONDITION

```
identifier is [NOT] { NUMERIC    }
                    { ALPHABETIC }
```

SIGN CONDITION

```
                                  { POSITIVE }
arithmetic-expression is [NOT]    { NEGATIVE }
                                  { ZERO     }
```

CONDITION-NAME CONDITION

condition-name

SWITCH-STATUS CONDITION

condition-nuns

NEGATED SIMPLE CONDITION

NOT simple-condition

COMBINED CONDITION

```
condition  { { AND }  condition }
           { { OR  }            }
```

ABBREVIATED COMBINED RELATION CONDITION

```
relation-condition  { { AND }  [NOT] [relational-operator] object } ...
                    { { OR  }                                     }
```

MISCELLANEOUS FORMATS

QUALIFICATION

```
{ data-name-1    }   [ { OF }  data-name-2 ]
{ condition-name }   [ { IN }              ]

paragraph-name  [ { OF }  section-name ]
                [ { IN }               ]

text-name  [ { OF }  library-name ]
           [ { IN }               ]
```

SUBSCRIPTING

```
{ data-name      }  (subscript-1 [, subscript-2 [, subscript-3]])
{ condition-name }
```

INDEXING

```
{ data-name      }  ( { index-name-1 [{±} literal-2] }
{ condition-name }    { literal-1                    }

     [ , { index-name-2 [{±} literal-4] }   [ , { index-name-3 [{±} literal-6] } ] ] )
     [   { literal-3                    }   [   { literal-5                    } ] ]
```

IDENTIFIER: FORMAT 1

```
data-name-1  [ { OF } data-name-2 ] ...  [ (subscript-1 [, subscript-2
               { IN }
    [, subscript-3]]) ]
```

IDENTIFIER: FORMAT 2

```
data-name-1  [ { OF } data-name-2 ] ...  [ ( { index-name-1 [{±} literal-2] }
               { IN }                        { literal-1                    }
    [ , { index-name-2 [{±} literal-4] }   [ , { index-name-3 [{±} literal-6] } ] ] ) ]
        { literal-3                    }       { literal-5                    }
```

GENERAL FORMAT FOR COPY STATEMENT

```
COPY text-name  [ { OF } library-name ]
                  { IN }
    [ REPLACING { { ==pseudo-text-1== }  BY  { ==pseudo-text-2== } } ... ]
                { { identifier-1      }      { identifier-2      } }
                { { literal-1         }      { literal-2         } }
                { { word-1            }      { word-2            } }
```

C
IBM AMERICAN NATIONAL STANDARD COBOL GLOSSARY

ACCESS: The manner in which files are referenced by the computer. Access can be sequential (records are referred to one after another in the order in which they appear on the file), or it can be random (the individual records can be referred to in a nonsequential manner).

Actual Decimal Point: The physical representation, using either of the decimal point characters (. or ,), of the decimal point position in a data item. When specified, it will appear in a printed report, and it requires an actual space in storage.

ACTUAL KEY: A key which can be directly used by the system to locate a logical record on a mass storage device.

Alphabetic Character: A character which is one of the 26 characters of the alphabet, or a space. In COBOL, the term does not include any other characters.

Alphanumeric Character: Any character in the computer's character set.

Alphanumeric Edited Character: A character within an alphanumeric character string which contains at least one B or 0.

Arithmetic Expression: A statement containing any combination of data-names, numeric literals, and figurative constants, joined together by one or more arithmetic operators in such a way that the statement as a whole can be reduced to a single numeric value.

Arithmetic Operator: A symbol (single character or two-character set) which directs the system to perform an arithmetic operation. The following list shows arithmetic operators:

Meaning	Symbol
Addition	+
Subtraction	
Multiplication	*
Division	/
Exponentiation	**

Assumed Decimal Point: A decimal point position which does not involve the existence of an actual character in a data item. It does not occupy an actual space in storage, but is used by the compiler to align a value properly for calculation.

BLOCK: *In COBOL, a group of characters or records which is treated as* an entity when moved into or out of the computer. The term is synonymous with the term Physical Record.

Buffer: A portion of main storage into which data is read or from which it is written.

Byte: A sequence of eight adjacent binary bits. When properly aligned, two bytes form a halfword, four bytes a fullword, and eight bytes a doubleword.

Channel: A device that directs the flow of information between the computer main storage and the input/output devices.

Character: One of a set of indivisible symbols that can be arranged in sequences to express information. These symbols include the letters A through Z, the decimal digits 0 through 9, punctuation symbols, and any other symbols which will be accepted by the data-processing system.

Character Set: All the valid COBOL characters. The complete set of 51 characters is listed in "Language Considerations."

Character String: A connected sequence of characters. All COBOL characters are valid.

Checkpoint: A reference point in a program at which information about the contents of core storage can be recorded so that, if necessary, the program can be restarted at an intermediate point.

Class Condition: A statement that the content of an item is wholly alphabetic or wholly numeric. It may be true or false.

Clause: A set of consecutive COBOL words whose purpose is to specify an attribute of an entry. There are three types of clauses: data, environment, and file.

COBOL Character: Any of the 51 valid characters (see CHARACTER) in the COBOL character set. The complete set is listed in "Language Considerations."

Collating Sequence: The arrangement of all valid characters in the order of their relative precedence. The collating sequence of a computer is part of the computer design -- each acceptable character has a predetermined place in the sequence. A collating sequence is used primarily in comparison operations.

COLUMN Clause: A COBOL clause used to identify a specific position within a report line.

Comment: An annotation in the Identification Division or Procedure Division of a COBOL source program. A comment is ignored by the compiler. As an IBM extension, comments may be included at any point in a COBOL source program.

Compile Time: The time during which a COBOL source program is translated by the COBOL compiler into a machine language object program.

Compiler: A program which translates a program written in a higher level language into a machine language object program.

Compiler Directing Statement: A COBOL statement which causes the compiler to take a specific action at compile time, rather than the object program to take a particular action at execution time.

Compound Condition: A statement that tests two or more relational expressions. It may be true or false.

Condition:

- One of a set of specified values a data item can assume.
- A simple conditional expression: relation condition, class condition, condition-name condition, sign condition, switch-status condition, NOT condition.

Conditional Statement: A statement which specifies that the truth value of a condition is to be determined, and that the subsequent action of the object program is dependent on this truth value.

Conditional Variable: A data item that can assume more than one value; one or more of the values it assumes has a condition-name assigned to it.

Condition Name: The name assigned to a specific value, set of values, or range of values, that a data item may assume.

Condition-name Condition: A statement that the value of a conditional variable is one of a set (or range) of values of a data item identified by a condition-name. The statement may be true or false.

CONFIGURATION SECTION: A section of the Environment Division of the COBOL program. It describes the overall specifications of computers.

Connective: A word or a punctuation character that does one of the following:

- Associates a data-name or paragraph-name with its qualifier
- Links two or more operands in a series
- Forms a conditional expression

CONSOLE: A COBOL mnemonic-name associated with the console typewriter.

Contiguous Items: Consecutive elementary or group items in the Data Division that have a definite relationship with each other.

Control Break: A recognition of a change in the contents of a control data item that governs a hierarchy.

Control Bytes: Bytes associated with a physical record that serve to identify the record and indicate its length, blocking factor, etc.

Control Data Item: A data item that is tested each time a report line is to be printed. If the value of the data item has changed, a control break occurs and special actions are performed before the line is printed.

CONTROL FOOTING: A report group that occurs at the end of the control group of which it is a member.

Control Group: An integral set of related data that is specifically associated with a control data item .

CONTROL HEADING: A report group that occurs at the beginning of the control group of which it is a member.

Control Hierarchy: A designated order of specific control data items. The highest level is the final control; the lowest level is the minor control.

Core Storage: Storage within the central processing unit of the computer, so called because this storage exists in the form of magnetic cores.

Cylinder Index: A higher level index, always present in indexed data organization. Its entries point to track indexes

Data Description Entry: An entry in the Data Division that is used to describe the characteristics of a data item. It consists of a level number, followed by an optional data-name, followed by data clauses that fully describe the format the data will take. An elementary data description entry (or item) cannot logically be subdivided further. A group data description entry (or item) is made up of a number of related group and/or elementary items.

DATA DIVISION: One of the four main component parts of a COBOL program. The Data Division describes the files to be used in the program and the records contained within the files. It also describes any internal Working-Storage records that will be needed (see "Data Division" for full details).

Data Item: A unit of recorded information that can be identified by a symbolic name or by a combination of names and subscripts. Elementary data items cannot logically be subdivided. A group data item is made up of logically related group and/or elementary items and can be a logical group within a record or can itself be a complete record.

Data-name: A name assigned by the programmer to a data item in a COBOL program. It must contain at least one alphabetic character.

DECLARATIVES: A set of one or more compiler-directing sections written at the beginning of the Procedure Division of a COBOL program. The first section is preceded by the header DECLARATIVES. The last section is followed by the header END DECLARATIVES. There are three options:

1. Input/output label handling
2. Input/output error-checking procedures
3. Report Writing procedures

Each has its standard format (see "Procedure Division").

Device-number: The reference number assigned to any external device.

Digit: Any of the numerals from 0 through 9. In COBOL, the term is not used in reference to any other symbol.

DIVISION: One of the four major portions of a COBOL program:

- IDENTIFICATION DIVISION, which names the program.
- ENVIRONMENT DIVISION, which indicates the machine equipment and equipment features to be used in the program.
- DATA DIVISION, which defines the nature and characteristics of data to be processed.
- PROCEDURE DIVISION, which consists of statements directing the processing of data in a specified manner at execution time.

Division Header: The COBOL words that indicate the beginning of a particular division of a COBOL program. The four division headers are:

- IDENTIFICATION DIVISION.
- ENVIRONMENT DIVISION.
- DATA DIVISION.
- PROCEDURE DIVISION.

Division-name: The name of one of the four divisions of a COBOL program.

EBCDIC Character: Any one of the symbols included in the eight-bit EBCDIC (Extended Binary-Coded-Decimal Interchange Code) set. All 51 COBOL characters are included.

Editing Character: A single character or a fixed two-character combination used to create proper formats for output reports (see "Language Considerations" for a complete list of editing characters).

Elementary Item: A data item that cannot logically be subdivided.

Entry: Any consecutive set of descriptive clauses terminated by a period, written in the Identification, Environment, or Procedure Divisions of a COBOL program.

Entry-name: A programmer-specified name that establishes an entry point into a COBOL subprogram.

ENVIRONMENT DIVISION: One of the four main component parts of a COBOL program. The Environment Division describes the computers upon which the source program is compiled and those on which the object program is executed, and provides a linkage between the logical concept of files and their records, and the physical aspects of the devices on which files are stored (see "Environment Division" for full details).

Execution Time: The time at which an object program actually performs the instructions coded in the Procedure Division, using the actual data provided.

Exponent: A number, indicating how many times another number (the base) is to be repeated as a factor. Positive exponents denote multiplication, negative exponents denote division, fractional exponents denote a root of a quantity. In COBOL, exponentiation is indicated with the symbol ** followed by the exponent.

F-mode Records: Records of a fixed length, each of which is wholly contained within a block. Blocks may contain more than one record.

Figurative Constant: A reserved word that represents a numeric value, a character, or a string of repeated values or characters. The word can be written in a COBOL program to represent the values or characters without being defined in the Data Division (see "Language Considerations" for a complete list).

FILE-CONTROL: The name and Header of an Environment Division paragraph in which the data files for a given source program are named and assigned to specific input/output devices.

File Description: An entry in the File Section of the Data Division that provides information about the identification and physical structure of a file.

File-name: A name assigned to a set of input data or output data. A file-name must include at least one alphabetic character.

FILE SECTION: A section of the Data Division that contains descriptions of all externally stored data (or files) used in a program. Such information is given in one or more file description entries.

Floating-Point Literal: A numeric literal whose value is expressed in floating-point notation -- that is, as a decimal number followed by an exponent which indicates the actual placement of the decimal point.

Function-name: A name, supplied by IBM, that identifies system logical units, printer and card punch control characters, and report codes. When a function-name is associated with a mnemonic-name in the Environment Division, the mnemonic-name can then be substituted in any format in which substitution is valid.

Group Item: A data item made up of a series of logically related elementary items. It can be part of a record or a complete record.

Header Label: A record that identifies the beginning of a physical file or a volume.

High-Order : The leftmost position in a string of characters.

IDENTIFICATION DIVISION: One of the four main component parts of a COBOL program. The Identification Division identifies the source program and the object program and, in addition, may include such documentation as the author's name, the installation where written, date written, etc. (see "Identification Division" for full details).

Identifier: A data-name, unique in itself, or made unique by the syntactically correct combination of qualifiers, subscripts, and/or indexes.

Imperative-Statement: A statement consisting of an imperative verb and its operands, which specifies that an action be taken, unconditionally. An imperative-statement may consist of a series of imperative-statements.

Index: A computer storage position or register, the contents of which identify a particular element in a table.

Index Data Item: A data item in which the contents of an index can be stored without conversion to subscript form.

Index-name: A name, given by the programmer, for an index of a specific table. An index-name must contain at least one alphabetic character. It is one word (4 bytes) in length.

Indexed Data-name: A data-name identifier which is subscripted with one or more index-names.

INPUT-OUTPUT SECTION: In the Environment Division, the section that names the files and external media needed by an object program. It also provides information required for the transmission and handling of data during the execution of an object program.

INPUT PROCEDURE: A set of statements that is executed each time a record is released to the sort file. Input procedures are optional; whether they are used or not depends upon the logic of the program.

Integer: A numeric data item or literal that does not include any character positions to the right of the decimal point, actual or assumed. Where the term "integer" appears in formats, "integer" must not be a numeric data item.

INVALID KEY Condition: A condition that may arise at execution time in which the value of a specific key associated with a mass storage file does not result in a correct reference to the file (see the READ, REWRITE, START, and WRITE statements for the specific error conditions involved).

I-O-CONTROL: The name, and the header, for an Environment Division paragraph in which object program requirements for specific input/output techniques are specified. These techniques include rerun checkpoints, sharing of same areas by several data files, and multiple file storage on a single tape device.

KEY: One or more data items, the contents of which identify the type or the location of a record, or the ordering of data.

Key Word: A reserved word whose employment is essential to the meaning and structure of a COBOL statement. In this manual, key words are indicated in the formats of statements by underscoring. Key words are included in the reserved word list.

Level Indicator: Two alphabetic characters that identify a specific type of file, or the highest position in a hierarchy. The level indicators are: FD, SD, RD.

Level Number: A numeric character or two-character set that identifies the properties of a data description entry. Level numbers 01 through 49 define group items, the highest level being identified as 01, and the subordinate data items within the hierarchy being identified with level numbers 02 through 49. Level numbers 66, 77, and 88 identify special properties of a data description entry in the Data Division.

Library-name: The name of a member of a data set containing COBOL entries, used with the COPY and BASIS statements.

LINKAGE SECTION: A section of the Data Division that describes data made available from another program.

Literal: A character string whose value is implicit in the characters themselves. The numeric literal 7 expresses the value 7, and the nonnumeric literal "CHARACTERS" expresses the value CHARACTERS.

Logical Operator: A COBOL word that defines the logical connections between relational operators. The three logical operators and their meanings are:

OR (logical inclusive -- either or both)

AND (logical connective -- both)

NOT (logical negation)

(See "Procedure Division" for a more detailed explanation.)

Logical Record: The most inclusive data item, identified by a level-01 entry. It consists of one or more related data items.

Low-Order: The rightmost position in a string of characters.

Main Program: The highest level COBOL program involved in a step. (Programs written in other languages that follow COBOL linkage conventions are considered COBOL programs in this sense.)

Mantissa: The decimal part of a logarithm. Therefore, the part of a floating-point number that is expressed as a decimal fraction.

Mass Storage: A storage medium -- disk, drum, or data cell -- in which data can be collected and maintained in a sequential, direct, or indexed organization.

Mass Storage File: A collection of records assigned to a mass storage device.

Mass Storage File Segment: A part of a mass storage file whose beginning and end are defined by the FILE-LIMIT clause in the Environment Division.

Master Index: The highest level index, which is optional, in the indexed data organization.

Mnemonic-name: A programmer-supplied word associated with a specific function-name in the Environment Division. It then may be written in place of the function-name in any format where such a substitution is valid.

MODE: The manner in which records of a file are accessed or processed.

Name: A word composed of not more than 30 characters, which defines a COBOL operand (see "Language Considerations" for a more complete discussion).

Noncontiguous Item: A data item in the Working-Storage Section of the Data Division which bears no relationship with other data items.

Nonnumeric Literal: A character string bounded by quotation marks, which means literally itself. For example, "CHARACTER" is the literal for and means CHARACTER. The string of characters may include any characters in the computer's set, with the exception of the quotation mark. Characters that are not COBOL characters may be included.

Numeric Character: A character that belongs to one of the set of digits 0 through 9.

Numeric Edited Character: A numeric character which is in such a form that it may be used in a printed output. It may consist of external decimal digits 0 through 9, the decimal point, commas, the dollar sign, etc., as the programmer wishes (see "Data Division" for a fuller explanation).

Numeric Item: An item whose description restricts its contents to a value represented by characters from the digits 0 through 9. The item may also contain a leading or trailing operational sign represented either as an overpunch or as a separate character.

Numeric Literal: A numeric character or string of characters whose value is implicit in the characters themselves. Thus, 777 is the literal as well as the value of the number 777.

OBJECT-COMPUTER: The name of an Environment Division paragraph in which the computer upon which the object program will be run is described.

Object Program: The set of machine language instructions that is the output from the compilation of a COBOL source program. The actual processing of data is done by the object program.

Object Time: The time during which an object program is executed.

Operand: The "object" of a verb or an operator. That is, the data or equipment governed or directed by a verb or operator.

Operational Sign: An algebraic sign associated with a numeric data item, which indicates whether the item is positive or negative.

Optional Word: A reserved word included in a specific format only to improve the readability of a COBOL statement. If the programmer wishes, optional words may be omitted.

OUTPUT PROCEDURE: A set of programmer-defined statements that is executed each time a sorted record is returned from the sort file. Output procedures are optional; whether they are used or not depends upon the logic of the program.

Overlay: The technique of repeatedly using the same areas of internal storage during different stages in processing a problem.

PAGE: A physical separation of continuous data in a report. The separation is based on internal requirements and/or the physical characteristics of the reporting medium.

PAGE FOOTING: A report group at the end of a report page which is printed before a page control break is executed.

PAGE HEADING: A report group printed at the beginning of a report page, after a page control break is executed.

Paragraph: A set of one or more COBOL sentences, making up a logical processing entity, and preceded by a paragraph-name or a paragraph header.

Paragraph Header: A word followed by a period that identifies and precedes all paragraphs in the Identification Division and Environment Division.

Paragraph-name: *A programmer-defined word that identifies and precedes a paragraph.*

Parameter: A variable that is given a specific value for a specific purpose or process. In COBOL, parameters are most often used to pass data values between calling and called programs.

Physical Record: A physical unit of data, synonymous with a block. It can be composed of a portion of one logical record, of one complete logical record, or of a group of logical records.

Print Group: An integral set of related data within a report.

Priority-Number: A number, ranging in value from 0 to 99, which classifies source program sections in the Procedure Division (see "Segmentation" for more information).

Procedure: One or more logically connected paragraphs or sections within the Procedure Division, which direct the computer to perform some action or series of related actions.

PROCEDURE DIVISION: One of the four main component parts of a COBOL program. The Procedure Division contains instructions for solving a problem. The Procedure Division may contain imperative-statements, conditional statements, paragraphs, procedures, and sections (see "Procedure Division" for full details).

Procedure-name: A word that precedes and identifies a procedure, used by the programmer to transfer control from one point of the program to another.

Process: Any operation or combination of operations on data.

Program-name: A word in the Identification Division that identifies a COBOL source program.

Punctuation Character: A comma, semicolon, period, quotation mark, left or right parenthesis, or a space.

Qualifier: A group data-name that is used to reference a non-unique data-name at a lower level in the same hierarchy, or a section-name that is used to reference a non-unique paragraph. In this way, the data-name or the paragraph-name can be made unique.

Random Access: An access mode in which specific logical records are obtained from, or placed into, a mass storage file in a nonsequential manner.

RECORD: A set of one or more related data items grouped for handling either internally or by the input/output systems (see "Logical Record").

Record Description: The total set of data description entries associated with a particular logical record.

Record-name: A data-name that identifies a logical record.

Reel: A module of external storage associated with a tape device.

Relation Character: A character that expresses a relationship between two operands. The following are COBOL relation characters:

Character	Meaning
>	Greater than
<	Less than
=	Equal to

Relation Condition: A statement that the value of an arithmetic expression or data item has a specific relationship to another arithmetic expression or data item. The statement may be true or false.

Relational Operator: A reserved word, or a group of reserved words, or a group of reserved words and relation characters. A relational operator plus programmer-defined operands make up a relational expression. A complete listing is given in "Procedure Division."

REPORT: A presentation of a set of processed data described in a Report File.

Report Description Entry: An entry in the Report Section of the Data Division that names and describes the format of a report to be produced.

Report File: A collection of records, produced by the Report Writer, that can be used to print a report in the desired format.

REPORT FOOTING: A report group that occurs, and is printed, only at the end of a report.

Report Group: A set of related data that makes up a logical entity in a report.

REPORT HEADING: A report group that occurs, and is printed, only at the beginning of a report.

Report Line: One row of printed characters in a report.

Report-name: A data-name that identifies a report.

REPORT SECTION: A section of the Data Division that contains one or more Report Description entries.

Reserved Word: A word used in a COBOL source program for syntactical purposes. It must not appear in a program as a user-defined operand.

Routine: A set of statements in a program that causes the computer to perform an operation or series of related operations.

Run Unit: A set of one or more object programs that function, at object time, as a unit to provide problem solutions. This compiler considers a run unit to be the highest level calling program plus all called subprograms.

S-mode Records: Records that span physical blocks. Records may be fixed or variable in length. Blocks may contain one or more segments. A segment may contain one record or a portion of a record. Each segment contains a segment-length field and a control field indicating whether or not it is the first and/or last or an intermediate segment of the record. Each block contains a block-length field.

SECTION: A logically related sequence of one or more paragraphs. A section must always be named.

Section Header: A combination of words that precedes and identifies each section in the Environment, Data, and Procedure Divisions.

Section-name: A word specified by the programmer that precedes and identifies a section in the Procedure Division.

sentence: A sequence of one or more statements, the last ending with a period followed by a space.

Separator: An optional word or character that improves readability.

Sequential Access: An access mode in which logical records are obtained from, or placed into, a file in such a way that each successive access to the file refers to the next subsequent logical record in the file. The order of the records is established by the programmer when creating the file.

Sequential Processing: The processing of logical records in the order in which records are accessed.

Sign Condition: A statement that the algebraic value of a data item is less than, equal to, or greater than zero. It may be true or false.

Simple Condition: An expression that can have two values, and causes the object program to select between alternate paths of control, depending on the value found. The expression can be either true or false.

Slack Bytes: Bytes inserted between data items or records to ensure correct alignment of some numeric items. Slack bytes contain no meaningful data. In some cases, they are inserted by the compiler; in others, it is the responsibility of the programmer to insert them. The SYNCHRONIZED clause instructs the compiler to insert slack bytes when they are needed for proper alignment. Slack bytes between records are inserted by the programmer.

Sort File: A collection of records that is sorted by a SORT statement. The sort file is created and used only while the sort function is operative.

Sort File Description Entry: An entry in the File Section of the Data Division that names and describes a collection of records that is used in a SORT statement.

Sort-file-name: A data-name that identifies a Sort File.

Sort-key: The field within a record on which a file is sorted.

Sort-work-file: A collection of records involved in the sorting operation as this collection exists on intermediate device(s).

SOURCE-COMPUTER: The name of an Environment Division paragraph. In it, the computer upon which the source program will be compiled is described.

Source Program: A problem-solving program written in COBOL.

Special Character: A character that is neither numeric nor alphabetic. Special characters in COBOL include the space ( ), the period(.), as well as the following:

+ - * / = $, ; ") (

SPECIAL-NAMES: The name of an Environment Division paragraph, and the paragraph itself, in which names supplied by IBM are related to mnemonic-names specified by the programmer. In addition, this paragraph can be used to exchange the functions of the comma and the period, or to specify a substitution character for the currency sign, in the PICTURE string.

Special Register: Compiler-generated storage areas primarily used to store information produced with the use of specific COBOL features. The special registers are: TALLY, LINE-COUNTER, PAGE-COUNTER, CURRENT-DATE, TIME-OF-DAY, COM-REG, SORT-RETURN, SORT-FILE-SIZE, SORT-CORE-SIZE, SORT-MODE-SIZE, and NSTD-REELS.

Standard Data Format: The concept of actual physical or logical record size in storage. The length in the Standard Data Format is expressed in the number of bytes a record occupies and not necessarily the number of characters, since some characters take up one full byte of storage and others take up less.

Statement: A syntactically valid combination of words and symbols written in the Procedure Division. A statement combines COBOL reserved words and programmer-defined operands.

Subject of entry: A data-name or reserved word that appears immediately after a level indicator or level number in a Data Division entry. It serves to reference the entry.

Subprogram: A COBOL program that is invoked by another COBOL program. (Programs written in other languages that follow COBOL linkage conventions are COBOL programs in this sense.)

Subscript: An integer or a variable whose value references a particular element in a table.

Switch-status Condition: A statement that an UPSI switch has been set to an ON or OFF condition. The statement may be true or false.

SYSIPT: The system input device.

SYSLST: The system output device.

SYSPCH: The system punch device.

SYSPUNCH: An alternate name for the system punch device.

System-name: A name, specified by IBM, that identifies any particular external device used with the computer, and characteristics of files contained within it.

Table: A collection and arrangement of data in a fixed form for ready reference. Such a collection follows some logical order, expressing particular values (functions) corresponding to other values (arguments) by which they are referenced.

Table Element: A data item that belongs to the set of repeated items comprising a table.

Test Condition: A statement that, taken as a whole, may be either true or false, depending on the circumstances existing at the time the expression is evaluated.

Trailer Label: A record that identifies the ending of a physical file or of a volume.

U-mode Records: Records of undefined length. They may be fixed or variable in length; there is only one record per block.

Unary Operator: An arithmetic operator (+ or -) that can precede a single variable, a literal, or a left parenthesis in an arithmetic expression. The plus sign multiplies the value by +1; the minus sign multiplies the value by -1.

UNIT: A module of external storage. Its dimensions are determined by IBM.

V-mode Records: Records of variable length, each of which is wholly contained within a block. Blocks may contain more than one record. Each record contains a record length field, and each block contains a block length field.

Variable: A data item whose value may be changed during execution of the object program.

Verb: A COBOL reserved word that expresses an action to be taken by a COBOL compiler or an object program.

Volume: A module of external storage. For tape devices it is a reel; for mass storage devices it is a unit.

Volume Switch Procedures: Standard procedures executed automatically when the end of a unit or reel has been reached before end-of-file has been reached.

Word:

1. In COBOL: A string of not more than 30 characters, chosen from the following: the letters A through Z, the digits 0 through 9, and the hyphen (-). The hyphen may not appear as either the first or last character.

2. In System/360: A fullword is 4 bytes of storage; a doubleword is 8 bytes of storage; a halfword is 2 bytes of storage.

Word Boundary: Any particular storage position at which data must be aligned for certain processing operations in System/360. The halfword boundary must be divisible by 2, the fullword boundary must be divisible by 4, the doubleword boundary must be divisible by 8.

WORKING-STORAGE SECTION: A section-name (and the section itself) in the Data Division. The section describes records and noncontiguous data items that are not part of external files, but are developed and processed internally. It also defines data items whose values are assigned in the source program.