POLISH-X TRANSFORMATIONS [1]

by
Lloyd D. Fosdick
Department of Computer Science
University of Colorado
Boulder CO 80309

CU-CS-203-81 28 May 1981

---

[1]Supported in part by NSF Grant MCS-8000017 and DOE Contract DE-AC02-80ER10718. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Abstract. This report describes rules for a component of TOOLPACK[COW79] which will transform FORTRAN 77 source programs into a standard format. These rules specify spacing of tokens, indentation of lines, numbering of labels, clarification of DO loops, etc. The objective of this tool is improved readibility of FORTRAN 77 programs.

1. Introduction.

POLISH-X is a software tool for formatting programs written in FORTRAN 77. It is similar to POLISH [DOR76], which was designed for FORTRAN 66 programs. The purpose of this report is to define the formatting transformations made by POLISH-X.

The choice of these transformations has been guided by the following general requirements.

(1) The formatted program must have the same computational characteristics as the unformatted program; that is, its input/output relations must be unaffected by formatting.

(2) The formatter must be efficient.

(3) The formatter must be easy to use.

(4) Transformations other than insertion and deletion of spaces are restricted to a small set: line break, label replacement, unique terminators for DO-ranges, movement of FORMAT statements, and insertion of identification information in columns 73-80.

The fourth of these requirements results from the notion that structural changes to the source text are the result of a functionally distinct set of source-to-source transformations which ought to be performed by a separate tool. However, a few structural transformations are included in POLISH-X because they are necessary, as in the case of line break, or because they seem to be particularly useful.

It is assumed that POLISH-X receives a stream of tokens created from the source text by a lexical analyzer and produces a sequence of lines, each line 80 characters in length. The input stream contains an end-of-statement token but no token to indicate the end of a line, and no token to indicate a space. Comment lines are indicated by tokens. Precise requirements on these tokens will be given in a separate report.

The rules for the appearance of lines produced by POLISH-X follow. Except where noted otherwise, the user cannot control a transformation.

2. Statement Margins.

Following the FORTRAN convention, character positions on a line are (implicitly) numbered 1, 2,..., 80, from left to right. Two positions, the left margin (LMARGS) and the

right margin (RMARGS) are distinguished. They must satisfy the following conditions:

LMARGS >= 7; RMARGS <= 72; RMARGS - LMARGS + 1 >= MINLEN

The parameters LMARGS, RMARGS, and MINLEN may be set by the user: they have the default values 7, 72, and 30, respectively.

3. Comment Lines.

A comment token represents a block of comment lines and the block's internal structure is preserved by all transformations. In particular, if there is indentation of a comment block then every line in the block is moved the same number of spaces, thus preserving relative indentation within the block.

The left margin (LMARGC) for comment lines is distinguished. The parameter LMARGC may be set by the user: it has the default value 3. If the user specifies LMARGC = 0, then all comment lines are left as they were in the source text.

The indentation of a comment block may be made to agree with indentation of a statement line in the sense that if a statement line is indented some number of spaces relative to LMARGS, then the comment block (if any) preceding the statement line is also indented the same number of spaces relative to LMARGC. This indentation of comment blocks is controlled by the parameter INDNTC which may be set by the user: if it is zero there is no indentation, otherwise there is; its default value is not zero. If LMARGC = 0, then INDNTC is ignored. When a comment block is indented the space between the column designated by LMARGC and the left margin of the indented comment block will be filled with the character {-}.

4. Spacing of Tokens.

In the following list the term "delimiter" means one of the symbols { ()'/ }. The apostrophe and slash serve as left and right delimiters. The interpretation of the slash as a delimiter is context dependent: it is a delimiter enclosing names of COMMON blocks, and enclosing lists of values in DATA statements.

The term operator has its FORTRAN 77 meaning except that it includes the assignment operator { = }. Precedence levels for operators are as specified in FORTRAN 77: the assignment operator is a lowest level binary operator.

The term "separator" means one of the symbols { ,: }.

In the following discussion the parameter PRNLVL is a non-negative integer denoting the depth of nesting of a token within parentheses: for a token not nested within parentheses PRNLVL=0. Comment lines are excluded from this discussion.

There is to be one space after tokens except as noted below.

* After left delimiter --- no space.

* Before right delimiter --- no space.

* Before separator --- no space.

* After separator --- if PRNLVL = 0 then one space else no space.

* After unary operator --- no space.

* Before and after binary arithmetic operator --- if PRNLVL = 0 and operator has lowest precedence then one space else no space.

* Before and after assignment operator --- if PRNLVL = 0 then one space else no space.

* Before and after relational operator --- if PRNLVL <= 1 then one space else no space.

* Before and after binary logical operator --- if PRNLVL <= 1 and operator has lowest, or next-to-lowest precedence, then one space else no space.

* Before and after concatenation operator --- no space.

* Otherwise, unless precluded by one of the above, in order of priority:

  ** Before left delimiter --- no space.

  ** After right delimiter --- if PRNLVL = 0 then one space else no space.

5. Indentation of Control Scope.

In FORTRAN 77 the word "range" is used to identify the control scope of a DO statement and the word "block" is used to identify the control scope of a block IF, ELSE, and, ELSE IF statement. Statements in a block or range are indented relative to their innermost control statement.

The rules for this indentation are as noted below.

* Statements in a DO range: indent INDNTS spaces;

* Statements in an IF-block: indent INDNTS spaces.

* Statements in an ELSE IF-block: indent INDNTS spaces.

* Statements in an ELSE-block:indent INDNTS spaces.

6. Breaking a Statement.

A statement must be broken, and a continuation line created, if the length of the statement is greater than RMARGS-ALMARG+1, where ALMARG is the actual left margin, as determined by the indentation. If no indentation is specified for the statement, then ALMARG = LMARGS. If the continuation line is too long then it must be broken, etc. The following rules control these actions.

* The length of a line followed by a continuation line is not less than (RMARGS-ALMARG+1)/2.

* The last token on a line followed by a continuation line is, in order of priority:

  ** a separator with PRNLVL = 0;

  ** a lowest precedence binary operator with PRNLVL = 0;

  ** a binary operator with PRNLVL = 0;

  ** a separator;

  ** a binary operator;

  ** any token except a left delimiter.

* If there is more than one choice at the same preference level for the last token, then select the rightmost.

* If no choice for the last token on a line is possible by these rules then ignore the first rule above (i.e. allow the line length to be less than (RMARGS-ALMARG+1)/2). If a line break still cannot be found then an error condition is signalled, the statement is truncated at column 72 and continued at column 7 of the next line.

* A continuation line is indented INDNTS spaces relative to the left margin of the initial line.

7. Blank Comment Line Before or After Some Statements.

A blank comment line appears before the following statements:

* ELSE

* ELSE IF

* first executable statement of program unit

* first FORMAT statement in a block of FORMAT statements

A blank comment line appears after the following statements:

* unconditional GO TO

* computed GO TO

* assigned GO TO

* RETURN

* STOP

* PAUSE

* END --- except for the last subprogram.

* last FORMAT statement in a block of FORMAT statements

These rules call for the insertion of a blank comment line only if a blank comment line is not already present.

8. Terminal Statement of a DO Loop.

The terminal statement of a DO-loop is a CONTINUE statement and it is not the terminal statement of another DO-loop. Note that this implies additional labels may be created.

9. Labels.

Labels are in increasing order with respect to their defining occurrence (appearance in columns 1-5) and are right justified. The first label has the value LABELS, subsequent labels increase in steps of LABELS. The default value of LABELS is 10.

10. FORMAT Statements.

The parameter MOVESF controls the placement of FORMAT statements in the text. If MOVESF=0 then FORMAT statements are not moved, otherwise they are moved to the end of the program unit in which they appear. The default value of MOVESF is 0.

11. Identification Field.

The user can request that line identification be placed in columns 73-80 with the parameter IDENTL. If IDENTL=0 then these columns will be filled with blanks, otherwise they will be filled as follows: columns 73-76 will contain the first four letters of the name of the program unit; columns 77-80 will contain the line number within the program unit, starting at 10 and increasing in steps of 10.

12. Key Words.

All key words are printed as they appear in the FORTRAN 77 Standards Document (ANSI X3.9-1978). Thus GOTO in the source text is printed as GO TO, etc.

13. Continuation Line Character.

This character is the currency symbol.

14. Embedded Stop-Start Control of Editing

If the source text contains the comment line

C$ STOP POLISH

then all text lines which follow will be reproduced in the source text with no alteration up to the first occurrence of the comment line

C$ START POLISH

These control lines are not reproduced in the output.

If labels are used in a segment of the source text delimited by C$ STOP... and C$ START... a warning is issued, since incorrect label referencing may result.

15. Error Handling.

If an error is detected in the source text, or formatting cannot continue for some other reason, the text of the program unit within which the interruption occurred is simply reproduced and a message identifying the cause of the formatting suspension is printed. Note that this suspension applies only to the unit (subroutine, function, main program, block data) in which the interruption occurred.

16. Summary of Transformations Under User Control. In the following list the name of the parameter is followed by its default value in parentheses, and then by an abbreviated descriptor.

LMARGS(7)....Left margin of statements.

RMARGS(72)...Right margin of statements.

MINLEN(30)...RMARGS-LMARGS+1 >= MINLEN.

LMARGC(3)....Left margin of comment text. LMARGC=0 means leave comment lines as is.

INDNTS(2)....Indentation for statement lines.

INDNTC(1)....Indentation for comment lines. INDNTC=1 means indent comment lines like statements lines.

MOVESF(0)....Move FORMAT statements. MOVESF=0 means don't move FORMAT statements.

LABELS(10)...First label and label increment.

IDENTL(1)....Identify lines in cols. 72-80. IDENTL=0 means don't identify lines.

17. References.

[COW79] Wayne R. Cowell, Webb C. Miller: The Toolpack Prospectus. Tech. Memo. 341, (Sep 1979), pp14. Applied Mathematics Division, Argonne National Laboratory, Argonne IL 60439.

[DOR76] John Dorrenbacher, David Paddock, David Wisneski, Lloyd D. Fosdick: POLISH, A FORTRAN Program to Edit FORTRAN Programs. Tech. Rept. 50, (May 1976), pp35. Dept. of Computer Science, University of Colorado, Boulder CO 80309.