Author(s) : M. Metcalf **Library:** PGMLIB
Submitter : **Submitted:** 01.02.1992
Language : Fortran **Revised:**

Fortran 77 to Fortran 90 source form conversion tool

Users of Fortran 90 can choose between two different styles of source form, the old (Fortran 77) and a new. This program reads code written according to the Fortran 77 *fixed* source form from a specified input file and writes it according to the Fortran 90 *free* source form to a specified output file. It also formats the code by indenting the bodies of `DO`-loops and `IF`-blocks, and performs a small number of syntax conversions.

Structure:

Complete `PROGRAM`
User Entry Names: `CONVERT`
Files Referenced: Input and output units, either default or user defined.

Usage:

`CONVERT` has the following calling sequence on all systems:

```
convert [-b] [-id n] [-il m] [-sb] ifile[.f] [ofile[.f90]]
        [+b]                 [+sb]
```

where the meaning of the arguments is as follows:

`-id` Indenting depth (default = `3`).

`-il` Maximal indenting level (default = `10`).

`-sb` Handle significant blanks (default).

`-b` Generate interface blocks only.

If no options are specified, significant blanks will be handled (`-sb`) and all code will be processed (`+b`). In order to do nothing but change the source form, type e.g.:

```
convert -id 0 -il 0 +sb mysource.f
```

Method:

The program converts between the old fixed Fortran 77 source form to the new Fortran 90 free source form. Note that blanks are significant in the new source form. In addition it is able to perform a few other useful operations on the fly.

Statement keywords are followed if necessary by a blank, and blanks within tokens are suppressed; this handling of blanks is optional, but the default (`-sb`).

If a `CONTINUE` statement terminates a single `DO` loop, it is replaced by `END DO`.

Procedure `END` statements have the procedure name added, if blanks are handled (`-sb`).

Statements like `INTEGER*2` are converted to `INTEGER(2)`, if blanks are handled (`-sb`). Depending on the target processor, a further global edit might be required (e.g. where 2 bytes correspond to `KIND=1`). Typed functions and assumed-length character specifications are treated similarly. The length specification `*4` is removed for all data types except `CHARACTER`, as is `*8` for `COMPLEX`. This treatment of non-standard type declarations includes any non-standard `IMPLICIT` statements.

Optionally, interface blocks only may be produced (`-b`); this requires blank processing to be requested (`-sb`). The interface blocks are written in a form compatible with both the old and the new source forms.

The program is able to handle Patchy Card files, as a `+` in column 1 is treated as a comment line

Restrictions:

The program does not indent `FORMAT` statements or any statement containing a character string with an embedded multiple blank. The order of comment lines and Fortran statements is slightly modified if there are sequences of more than 200 comment lines. If there are syntax errors, continued lines do not have a trailing `&`.

When producing interface blocks, a check is required that any dummy argument that is a procedure has a corresponding `EXTERNAL` statement. Also, since no `COMMON` blocks or `PARAMETER` statements are copied, part of an assumed-size array declaration may be missing. Similarly, parts of an assumed-length character symbolic constant might be copied and have to be deleted. `BLOCK DATA` statements are copied and must be deleted. These problems would normally be detected by a compiler and are trivially corrected.

Within a given keyword, the case must be all upper or all lower, and lower case programs require blank handling for correct indenting.

Error handling:

Primitive syntax checks protect the program from most non-Fortran source input.

References:

1. M. Metcalf and J.Reid, Fortran 90 explained, Oxford Science Publications (1990), Chapter 2

•