RATFOR _ A Rational Fortran

B. W. Kernighan

Bell Laboratories
Murray Hill, N. J. 07974

Fortran programs are hard to read, write and debug. To make program development easier, RATFOR provides a set of decent control structures:

statement grouping
completely general **if - else** statements
while, **for** and **do** for looping
break and **next** for controlling loop exits

and some ‘‘syntactic sugar’’:

free form input (e.g., multiple statements/line)
unobtrusive comment convention
translation of >, >=, etc., into .GT., .GE., etc.
‘‘define’’ statement for symbolic parameters
‘‘include’’ statement for including source files

RATFOR not only makes programming in Fortran more enjoyable, but also allows structured programming, in the sense of coding without GOTO’s. RATFOR programs tend to be markedly easier to write, read, and make correct than their equivalents in standard Fortran.

RATFOR is a preprocessor, translating the input into standard Fortran constructions. RATFOR programs may readily be written so the generated Fortran is portable; program transferability is easy to achieve. RATFOR is written in itself, so it is also portable.

The grammar of RATFOR is as follows:

```
prog  : stat
      | prog  stat
stat  : if( condition ) stat
      | if( condition ) stat else stat
      | while( condition ) stat
      | for( initialization; condition; increment ) stat
      | do do-part stat
      | break
      | next
      | digits  stat
      | { prog }
      | anything unrecognizable
```

In the grammar above, condition can be any legal Fortran condition like "‘A .EQ. B’," i.e., anything that could appear in a legal Fortran logical IF statement. stat is, of course, any Fortran or RATFOR statement, or any collection of these enclosed in braces.

The while statement performs a loop while some specified condition is true. The test is performed at the *beginning* of the loop, so it is possible to do a while zero times, which can't be done with a Fortran DO.

The for statement is a somewhat generalized while statement that allows an initialization and an incrementing step as well as a termination condition on a loop. initialization is any single *Fortran* statement, which gets done once before the loop begins. increment is any single *Fortran* statement, which gets done at the end of each pass through the loop, before the test.

for and while are useful for backward loops, chaining along lists, loops that must be done zero times, and similar things which are hard to express with a DO statement, and obscure to write out directly.

The do statement is like a Fortran DO, except that no label or CONTINUE is needed. The **do-part** that follows the do keyword has to be something that can legally go into a Fortran DO statement.

A break causes an immediate exit from a for, while or do to the following statement. The next statement causes an immediate transfer to the increment part of a for or do, and the test part of a while. Both break and next refer to the innermost enclosing for, while or do.

Statements can be placed anywhere on a line; long statements are continued automatically. Multiple statements may appear on one line, if they are separated by semicolons. No semi-colon is needed at the end of a line, if RATFOR can guess whether the statement ends there. Lines ending with any characters obviously a continuation, like plus or comma, are assumed to be continued on the next line. Any statement that begins with an all-numeric field is assumed to be a Fortran label, and placed in columns 1-5. PP A '#' character in a line marks the beginning of a comment; the comment is terminated by the end of a line.

Text enclosed in matching single or double quotes is converted to nH... by RATFOR, but is otherwise unaltered. Characters like '>', '>=', and '&' are translated into their longer Fortran equivalents '.GT.', '.GE', and '.AND', except within quotes.

Any string of alphanumeric characters can be defined as a name; thereafter, whenever that name occurs in the input (delimited by non-alphanumerics) it is replaced by the rest of the definition line (comments are stripped off).

An entire source file may be included by saying "include filename" at the appropriate place.