1-2 COMPARISON OF FORTRAN AND C

****

(Thanks to Craig Burley for the excellent comments)

The world of computing sometimes adopts silly fashions, too often good companies and products fell from grace, and lesser ones gained the upper hand. Some new examples for the uselessness of quality are the MS empire and Compaq buying Digital Equipment Corporation.

It seems that the fashion winds (in the US, in the UK it seems to be different) blows now in the numerical computing world towards C and C++. This strange trend is probably driven by people who are not experienced numerical programmers.

Dr. John Prentice
Fortran 90 as a language of choice for science students
At Lahey
At UCD

Jerrold Wagener
Fortran 90 and Computational Science
Was available at the CSEP website

Fortran still dominates in the numerical computing world, but it seems to lose ground. The following points may help you make up your mind.

(Partly adapted from the Fortran FAQ)

a) FORTRAN tends to meet some of the needs of scientists better. Most notably, it has built in support for:

- Variable-dimension array arguments in subroutines. A feature required for writing general purpose routines without explicitly specifying the array dimensions passed to them. Standard C lacks this important feature (some compilers like gcc have it as non-standard extension) and the workarounds are very cumbersome (See Appendix C).

  This feature by itself is sufficient to prefer Fortran over C in numerical computing.

- A rich set of useful generic-precision intrinsic functions. Such functions can be highly optimized (written in assembly language with optimized cache utilization), and they make programs standard at a higher level (and more portable).

- Builtin complex arithmetic (arithmetic involving complex numbers represented as having real and imaginary components).

- Array index-ranges may start and end at an arbitrary integer, the C convention of [0,N-1] is usually inconvenient.

- Better I/O routines, e.g. the implied do facility gives flexibility that C's standard library can't match. The Fortran compiler directly handles the more complex

syntax involved, and as such syntax can't be easily reduced to argument passing form, C can't implement it efficiently.

- o A compiler-supported infix exponentiation operator which is generic with respect to both precision and type, AND which is generally handled very efficiently, including the commonly occurring special case floating-point**small-integer.

- o Fortran 90 supports an array notation that allows operations on array sections, and using vector indices.

  The new intrinsic functions allow very sophisticated array manipulations.

  The new array features are suitable for parallel processing.

- o Fortran 90 supports automatic selection of numeric data types having a specified precision and range, and makes Fortran programs even more portable.

- o Fortran extensions for parallel programming are standardized by the High Performance Fortran (HPF) consortium.

Fortran 90 supports useful features of C (column independent code, pointers, dynamic memory allocation, etc) and C++ (operator overloading, primitive objects).

b) The design of FORTRAN allows maximal speed of execution:

- o FORTRAN 77 lacks explicit pointers, which is one reason that it is more amenable to automatic code optimization. This is very important for high-performance computing.

  Fortran 90 allows explicit pointers restricted to point only to variables declared with the "target" attribute, thus facilitating automatic optimizations.

- o Fortran was designed to permit static storage allocation, saving the time spent on creating and destroying activation records on the stack every procedure call/return.

  Recursive procedures are impossible with static allocation, but can be simulated efficiently when needed (very rare).

- o Fortran implementations may pass all variables by reference, the fastest method.

- o Fortran disallows aliasing of arguments in procedure-call statements (CALL statements and FUNCTION references), all passed argument lists must have distinct entries.

  Fortran disallows also aliasing between COMMON (global) variables and dummy arguments.

  These restrictions allows better compiler optimizations.

c) There is a vast body of existing FORTRAN code (much of which is

publicly available and of high quality). Numerical codes are particularly difficult to port, scientific establishments usually do not have large otherwise idle programming staffs, etc. so massive recoding into any new language is typically resisted quite strongly.

d) FORTRAN 77 tends to be easier for non-experts to learn than C, because its 'mental model of the computer' is much simpler.

For example, in FORTRAN 77 the programmer can generally avoid learning about pointers and memory addresses, while these are essential in C. More generally, in FORTRAN 77 the difference between (C notation) x, &x, and often even *x is basically hidden, while in C it's exposed. Consequently, FORTRAN 77 is a much simpler language for people who are not experts at computer internals.

Because of this relative simplicity, for simple programming tasks which fall within its domain, (say writing a simple least-squares fitting routine), FORTRAN 77 generally requires much less computer science knowledge of the programmer than C does, and is thus much easier to use.

Fortran 90 changes the picture somewhat, the new language is very rich and complex, but you don't have to use or even know about all this complexity.

e) The C standard requires only a basic double-precision mathematical library, and this is often what you get. The FORTRAN standard, on the other hand, requires single & double precision math, many vendors add quad-precision (long double, REAL*16) and provide serious math support.

Single-precision calculations may be faster than double-precision calculation even on machines where the individual machine instructions takes about the same time because single-precision data is smaller and so there are less 'memory cache misses'.

Quad-precision (long double) calculations are sometimes necessary to minimize roundoff errors.

If you have only double-precision mathematical routines, the basic mathematical primitives will take up unnecessary CPU time when used in single-precision calculations and will be inexact if used with 'long double'.

f) FORTRAN is designed to make numerical computation easy, robust and well-defined:

1) The order of evaluation of arithmetical expressions is defined precisely, and can be controlled with parentheses.

2) The implicit type declaration feature saves time/typing
   (however it makes your program vulnerable to annoying
   and hard to detect bugs).

3) Case insensitivity eliminates bugs due to 'miscased'
   identifiers.

4) The lack of reserved words in the language gives the
   programmer complete freedom to choose identifiers.

5) The one statement per line principle (of course
   continuation lines are allowed with a special syntax)
   makes programs more robust.

6) Added blanks (space characters) are insignificant
   (except in character constants) this also contributes
   to the robustness of FORTRAN programs.

7) Linking with the mathematical library doesn't require
   any compiler option (in C you to have to use "-lm").

g) Last but not least, FORTRAN compilers usually emit much better
   diagnostic messages.

In summary, we can say that the difference between Fortran and C,
is the difference between a language designed for numerical computations,
and a language designed for other purposes (system programming).

```
+-------------------------------------------+
|                                           |
|     SUMMARY OF FORTRAN ADVANTAGES         |
|     ============================          |
|     a) Scientifically oriented            |
|     b) Better optimized code              |
|     c) A lot of existing code             |
|     d) Easier to learn                    |
|     e) More efficient mathematics         |
|     f) Easier to use and more robust      |
|     g) Better diagnostics                 |
|                                           |
+-------------------------------------------+
```