Comments on benchmarks

Editor:
“Selecting a Programming Language, Compiler, and Support Environment: Method and Example” in the August 1982 issue does a disservice to the computer community. Benchmarks are bad enough—even if published in their entirety—since they invariably are not decoded by experts in their respective languages. In this case the benchmarks algorithm was not even published, so the various timing figures in Table I could not be verified by a disinterested party.

For example, a language like Forth should not be coded in Pascal’s footprint. Translating a Pascal program directly to Forth is generally a good way to produce inefficient code. Most programmers don’t realize to what extent their problem-analysis thinking processes are colored by the language they generally use for coding. Forth is philosophically different enough from other languages so that a certain amount of unlearning is usually required before a programmer can become proficient in it.

Furthermore, although the authors did not intend that Table I be used as a general evaluation of these languages, the temptation to do so will prove overwhelming. Even they are fooled by their own benchmark when they say, “As might have been expected, Fortran, the oldest language evaluated, was last.”

Gary Feierbach
Inner Access Corp.
Belmont, Calif.

Authors’ reply

Mr. Feierbach states that benchmarks are bad because they are not coded by experts in their respective languages. Benchmarks should not necessarily be coded by experts because production code is not always written by experts. Nevertheless, the benchmarks used in the study were coded by experienced computer scientists who had programmed in a number of different programming languages.

The letter also states, “Forth should not be coded in Pascal’s footprint.” We agree. Programming in Forth is quite different from programming in Pascal, C, or Fortran. Forth’s incompatibility with a Pascal-like style highlights its unusual nature. In our study, Forth was eliminated because it ran slowly under the preferred development system, CP/M. However, Forth probably should have been eliminated from consideration earlier, at the same stage as APL, Cobol, and Lisp. Forth’s domain of programming applicability does not include large software systems requiring substantial software maintenance.

Remember, our evaluation chose a language for a particular application, one that will consist of large programs developed by large programming teams and that will require long-term software maintenance. There are applications where Forth might be the best choice: for example, a small program, written by one person, for which few changes are anticipated.

In regard to the benchmark results (Table I in the article) being used for general evaluation of these languages, we must state once again that the main point of the article was the methodology, not the specific rankings of the languages. The scores are dependent upon the programming application for two reasons. First, the reviewers were given direction to consider the application both when determining the relative weights for the technical criteria and when assigning scores to each language for each criteria. Second, as explained in the article, the benchmark was carefully adjusted to reflect the relative mix of expected processing for the specific communications application. Given these factors, we still fail to be surprised that Fortran received low scores. Forth was last overall primarily because of its lack of effective control logic and data structures.

Another reason for not attempting to use the benchmark results as a general evaluation of these languages is that they are now old data. Compilers have been improved, new compilers have been developed, usage characteristics have changed, and other factors affecting the scores on the technical criteria are no longer the same. To now use these benchmark results as the sole reason for selecting a language would be to commit the very error we hoped to counsel against. Selecting a programming language, compiler, and support environment is a difficult and important task and deserves the application of the methods we have suggested.

For those who wish additional detail on how the methodology was applied, a government report on the study is available from the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161.

The title of the report is Digital Communications Terminal High Order Programming Language Study (DCHOL Study), Volumes I and II, 26 November 1980. Among other detail and background material on why the study was conducted, the report contains specific information on language scores for each technical feature and the benchmark source code of each language for which figures of merit were calculated.

Gordon E. Anderson
Kenneth E. Shumate

To be considered for publication, a letter to the editor must be accompanied by a statement giving Computer permission to publish that letter.

November 1982