"On selecting an implementation language: C or Pascal?" draws comments

My first review column—"On Selecting an Implementation Language: C or Pascal?"—generated a number of comments. In this issue I share two of the more extensive and penetrating comments I received. They bring up several ideas that I overlooked in my original evaluation of C and Pascal.

In addition, Jonathan Sachs of Lotus Development Corp. wrote to reprimand me for an error: "...you state that 1-2-3 is written in C. In actuality, it is written in 8086 assembly language." I stand corrected.

James L. Peterson of Carnegie-Mellon University responded to the comparison of the two languages with an interesting observation. He suggests that C and Pascal can be used together as an implementation language: Pascal is used as a "checkout compiler," and C is used to gain speed and portability. Actually, I am aware of several Pascal-to-C translators around that convert from a specific dialect of Pascal (say, UCSD Pascal) to K&R standard C. But Peterson expresses the technique very well.

K. Ramesh Babu of Excelan, Inc., writes, '"...C invites the disciplined programmer to design more appropriate abstractions than may be (pre-)determined in a high-level language." I think his observation is significant because it points up the desire for the flexibility offered by a low-level language. It would be a mistake to think that C programmers are attracted to C because it is less disciplined or lower leveled. Instead, Babu suggests that the attraction lies in the freedom to choose one's own high-level structures. He does not advocate either C or Pascal, but his point is well taken: abstraction is achievable in C without the overhead or restrictions of the compiler.

T.G.L.

Translating Pascal into C

As a matter of course, I have been doing all of my programming in Pascal, both because it is available on all of the computer systems that I use, and because I find Pascal to be the clearest language in which to write. However, I know of no Pascal compiler that has been specifically designed to generate high-quality object code. Most Pascal compilers stress other aspects of compiler design, such as error recovery or portability, or were academic projects.

C, on the other hand, while not nearly so easy to read or understand, was designed, in part, to allow good object code generation. C compilers tend to be particularly good on Unix systems, where the operating system itself is written in C.

There is no intrinsic reason why the code generated by a C compiler would necessarily be better than the code produced by a Pascal compiler. However, in practice, we would expect the same algorithm, written in C and in Pascal, to execute faster in C than in Pascal. The C compiler will probably produce better object code than the Pascal compiler. A reasonable approach for production programs might then be to develop the program in Pascal, for readability and ease of debugging, and then translate the Pascal source into C source for repeated execution.

A Pascal to C translator. Pascal and C are quite similar languages. Both are block-structured procedural languages. Pascal is mostly a subset of C, so translation from Pascal to C is relatively easy. C has a number of operators and structures that have no equivalent in Pascal (such as conditional expressions or the break statement), so translation from C to Pascal would be considerably more difficult.

A reasonable text editor allows translation of a small program of a few hundred lines of code from Pascal to C in an hour or so. I translated one program, which I use often, from Pascal to C by hand. It was a program that selected fields from a 20-megabyte data file on a Vax for later processing. Running the Pascal version took about 50 minutes, while the C version of the same program took about five minutes. This led me to believe that translating Pascal programs to C might lead to significantly faster processing, presumably because of the better C compiler.

In fact, a program can be written to translate Pascal to C (mostly) automatically. A design study was done by Nancy Springer, and a first translator was then written by Gad Dafni at the University of Texas at Austin. I have since rewritten nearly all of the translator. The translator was, of course, written in Pascal and successfully translates itself into C.

The translator uses a top-down recursive descent compilation method to recognize the input Pascal program. For each Pascal construct, the corresponding parsing routine generates an output string that is the equivalent C source code. Most Pascal constructs have a fairly obvious C equivalent, and so the conversion is straightforward.

For example, the Pascal

```pascal
if boolean-expression
  then statement1;
else statement2;
```

is translated to the C

```c
if (boolean-expression)
  statement1;
else statement2;
```

A major aim of the translation is to retain the original flavor of the Pascal program. Thus, although we could replace symbolic constants with their values, it seems more appropriate to retain them as symbolic constants in the C program. The output of the translator has the same structure, statements and variables as the input—only the language (and syntax) of expression has been changed.