Bad Code, Good Code

by Frederick L. Hiltz, Stage Manager

Just Ask! always has been written anonymously. I undertake the job with much respect for my backstage predecessor, who established a high standard to follow. Like the Stage Manager in Thornton Wilder's *Our Town*, I expect to set the players in their positions, offer some introductory and explanatory comments, and occasionally step into the action. You are the players. Do you have a question that deserves discussion? Have you found a good answer to someone else's question that you would like to share? How about a controversial question and a discussion of pros and cons? If you prefer that your name not be published, please say so in your contribution, which should be sent to the managing editor at *M Computing*.

Question: How should I recognize good code when I see it? What are the hallmarks of good programming techniques?

C. Greenock sends a fine example contrasting bad code that "works" and good code that also works:

It is not uncommon to have to write searches to allow a user to enter some or all of the search parameters. This can lead to some quite convoluted code when attempting to order down a global where a search key may be null or not.

For example, a user attempting to find a patient may enter any or all of surname, forename, middle name, and date of birth. We'll assume for the sake of argument that we aren't interested in partial matches and that the index global is structured as shown here:

```
^ix(surname,forename,middlename,
 dob,integer)=patID
```

The original solution was structured roughly as shown in figure 1. Horrible, isn't it? And this was an attempt to avoid spaghetti code. It worked but was unsatisfactory.

```
      ;
FORE FOR       DO QUIT:(END!(FORE=""))
     .; We have a known key so drop to next level
     .IF SURN'="" DO MID
     .; Otherwise retrieve key.
     .IF 'END SET FORE=$ORDER(^ix(SURN,FORE)) QUIT:FORE=""
     QUIT
     ;
MID  FOR  DO  QUIT:(END!(FORE=""))
     .IF FORE'="" D DOB
     .IF 'END SET MID=$ORDER(^ix(SURN,FORE,MID)) QUIT:MID=""
     QUIT
     ;
DOB  ; similar code to above and determine end condition
```

Figure 1. Sample of unsatisfactory code, although it works.

The code in figure 2 achieves the same ends as that in figure 1, but a great deal more compactly and more obviously. If we are not looking for partial matches (i.e., not aiming to retrieve WILLIAMSON as well as WILLIAMS on a search for a surname WILLIAM) then the code requires no further modification.

```
DO  ; start search
.;  p1      p2          p3
.DO:SU'="" QUIT:SU'=""  FOR  SET SU=$ORDER(^ix(SU)) QUIT:SU=""  DO
..DO FO'="" QUIT:FO'=""  FOR  SET FO=$ORDER(^ix(SU,FO)) QUIT:FO=""  DO
...DO MID'="" QUIT:MID'=""  FOR  SET MID=$ORDER(^ix(SU,FO,MID)) QUIT:MID=""  DO
....DO:DOB'="" QUIT:DOB'=""  FOR  SET DOB'$ORDER(^ix(SU,FO,MID,DOB)) QUIT:DOB=""  DO
.....DO WHATEVER
;
```

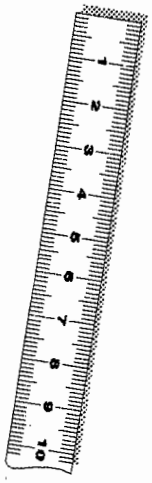
Figure 2. Successful code.

Examining line `.DO:SU'=""` . . . we see that if a non-null surname (SU) is supplied that control will be passed to the next line `..DO FO'=""` . . . from position p1 (as marked in the comment above); otherwise a `$ORDER()` is performed at position p3 to obtain the first surname on the index and control then moves to `..DO FO'=""`. When the code at `..DO FO'=" "` passes control back to the line above, it will either pass control back to position p2 (if `..DO FO'=""` was invoked from position p1) or to position p3 if `..DO FO'=""` . . . was called from position p3.

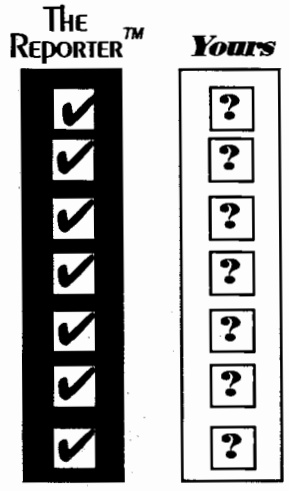
Notice the indent at the first line of figure 2. This is required because, in the case of a known surname, the routine would quit after the code in the nest had executed. An alternative, to reduce the level of nesting, would be to place the search in a subroutine.

A Report Generator That Really Measures Up!

THE REPORTER™ ~ *"the most powerful and easy to use report writer in the market today..."*



1. Windows-based .....
2. Does cross-tab reports .....
3. Easily adds graphics and full-featured charts.....
4. Produces WYSIWYG reports .....
5. Connects to all popular databases including M.....
6. Exports to all popular Windows applications .....
7. Includes database- and report-specific security.....



If your report generator doesn't measure up, call us.

Sea Change Systems Inc. **(800)-535-4585**

3 CENTENNIAL DRIVE, SUITE G. □ PEABODY, MA 01960

A useful variant of this idea, given in figure 3, makes coding of "interruptible" searches/processes a little easier. Care has to be taken to ensure that the appropriate keys are cached to avoid repeating PROCESS for the last data retrieved.

```
        ; Retrieve keys from cache, returned null if start of search.
        DO GETKEYS(.K1,.K2,.K3)
        SET PAUSE=0
SRCH    DO
        .DO:K1'=""  FOR  SET K1=$ORDER(^glob(K1)) QUIT:K1=""  DO  QUIT:PAUSE
        ..DO:K2'=""  FOR  SET K2=$ORDER(^glob(K1,K2)) QUIT:K2=""  DO  QUIT:PAUSE
        ...DO:K3'=""  FOR  SET K3=$ORDER(^glob(K1,K2,K3)) QUIT:K3=""  DO  QUIT:PAUSE
        ....SET PAUSE=$$PAUSE IF PAUSE DO SAVEKEYS(K1,K2,K3) QUIT
        ....; Examine pause flag before processing to ensure that PROCESS is
        ....; not run twice over the data represented by K1,K2,K3.
        ....DO PROCESS
        ;
        ; Subsequent code.
```

Figure 3. Code variation.

Stage Manager: Figure 2 is easily recognized as better code than figure 1. Why? The most important purpose of a program is to communicate a method from one person to another, right? When the next person reads this program to extend it or adapt it, several things about figure 2 promote that communication:

- The structure of the method is clear at a glance. How long would you study figure 1 to discover the nested loops?
- Figure 2 contains no labels. We can read it from the top down, confident that it will not be entered in the middle. Good code reserves labels for reusable subroutines and functions—a subroutine called from only one place usually indicates a structure problem.

  GOTO destroys more structures than any other command. Good code uses it only to create structures not available directly in the language, a truly rare requirement. When GOTO disappears, so do many labels.
- "Flag" variables often signify bad structure. Figure 1 needs the END variable for that reason. Ask of every variable, "Does it pertain to the application, or does it merely serve this program?" If the latter, then a better structure is almost always available.

Frederick L. Hiltz, Ph.D., develops medical information system software at Brigham and Women's Hospital, Boston, Massachusetts.

C. Greenock, a senior analyst/programmer with SWift Information Technology in Bristol, England, contributed the substance of this *Just Ask!* column.