FURTHER COMMENTS ON THE PREMATURE LOOP EXIT PROBLEM

C.K. YUEN
COMPUTER CENTRE
UNIVERSITY OF HONG KONG *

Following the distribution of my earlier contribution to SIGPLAN Notices[1] several collegues commented to me that they can produce better code for the "premature loop exit" program than what were shown in the article and that my discussion was too lenient to GOTO statements. Since such a reaction is likely to be quite common among the readers of this publication, I feel it would be useful to present a few brief comments here that serve to highlight the main thrust of the article.

The first solution presented to me is as follows:

```
STAYINLOOP := TRUE; I := 1;
WHILE STAYINLOOP DO
BEGIN
     ...
     IF A THEN
     BEGIN
          STAYINLOOP := FALSE:
          X;
     END ELSE
     BEGIN
          ...
          IF B THEN
          BEGIN
               STAYINLOOP := FALSE;
               Y;
          END ELSE
          BEGIN
               I := I+1;
               IF (I>N) THEN
               BEGIN
                    STAYINLOOP := FALSE;
                    Z;
               END
          END
     END
END
```

As is required, the program will enter the loop, carry out some processing and then test for condition A, whose presence causes loop exit to perform action X. In its absence further processing and testing for condition B occur, possibly leading to loop exit and action Y. After N repetitions without A or B occurring loop termination takes place with action Z.

However, I believe a better solution is the following:

Present address: Dept of Computer Science, National University of Singapore, Kent Ridge, Singapore 0511.

```
CONTROL := 0; I := 1;
WHILE (CONTROL=0) DO
BEGIN
     ...
     IF A THEN CONTROL := 1 ELSE
     BEGIN
          ...
          IF B THEN CONTROL := 2 ELSE
          BEGIN
               I := I+1;
               IF (I>N) THEN CONTROL := 3;
          END
     END
END
CASE CONTROL OF
     1 : X;
     2 : Y;
     3 : Z;
END
```

For, in solution 2 the text of the program fragment indicates very clearly the three loop exit conditions and the alternative actions they lead to out of the loop. In the case of solution 1, it takes a bit of mental effort to realize that actions X, Y or Z are not repeated despite their inclusion inside the loop.

It will be quite fair to conclude from the above discussion that the maxim "we can do anything you do with GOTOs without GOTOs" has been reaffirmed yet again. Satisfying as this might be, I think it would be more fruitful to return to the main point of the earlier article in light of the new examples:

If a program is to be easily comprehensible, then it needs to display in a fairly obvious fashion the links between the occurrences of conditions and their consequent actions. Such links can be provided in a variety of ways, and our aim should be to employ program constructs whose form matches the underlying "topological" structure. Solution 2 above works well because of just that: CONTROL, the "link" used here, has four alternative values which, when used in conjunction with the WHILE and the CASE, selects four alternative actions (stay in loop, X, Y or Z). Because of the good structural match, it does not matter very much that the values are themselves not meaningful. (There would have been little improvement to make CONTROL a special ordinal variable with, say, values like (STAYINLOOP, ACTIONX, etc), though in a larger loop this might be of some value.)

So we need not be surprised that GOTO statements, which can be quite effective for the purpose of creating a simple point to point link in the program, are very bad for implementing complex structures, just as Boolean control variables can be overused and produce a mess. The question is not whether the "geometric" structures are themselves good or bad; it is how well they match the required topology.

[1] C.K. Yuen, "The programmer as navigator", SIGPLAN Notices, 18,9 (Sept. 1983), p.70