SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 9, 821–825 (1979)

Pascal Stylistics and Reserved Words

ARTHUR SALE
Department of Information Science, University of Tasmania, Hobart, Tasmania, Australia

May I disagree strongly with Dr. Bishop's paper "*On publication Pascal*",[1] published recently in this journal? It is difficult to pinpoint exactly where we part company, but I was horrified to read the conclusions: that keywords should be capitalized in Pascal programs as a preferred style. As it happens, I have been trying to discourage this practice for some time now. Since it is difficult to put the alternative case by reference to Dr Bishop's paper (I need to question some basic assumptions), I shall start from scratch.

SHOULD PASCAL KEYWORDS BE DISTINGUISHED IN STYLE?

In brief, my view is that consistently distinguishing keywords by capitalizing them in contrast to predominantly lower-case identifiers is an infantile phase that we should have grown out of. Consider how we write English: we use predominantly lower-case letters, with first letters of sentences and proper nouns capitalized to mark them, and the occasional use of capitals for EMPHASIS. We are generally better readers of this style of text than of any other. Since the readability of Pascal text is high, and it has a small set of special words which can be easily remembered, there is no good reason for singling out the reserved words for special treatment. Consequently, I argue that the most desirable style to standardize is one where reserved words are in lower-case letters and undistinguished from the rest of the program text. For example:

```
program mean(input, output);
  var
    value, sum: real;
    count: integer;
  begin
    sum:=0;
    ...
```

I remember as a child learning to read with some books in which the difficult words (the new ones I was learning) were picked out in red. I grew out of that a long time ago. The situation with respect to Pascal reserved words is not much different.

In textbooks intended for people unfamiliar with the language, there is a case for making *minor* distinguishing features to assist the learning process. Any such distinguishing of reserved words must not disrupt the normal flow of reading. For example, the following distinguishing features may be used, listed in order of preference:

0038-0644/79/1009-0821$01.00 *Received 16 February 1979*

distinguishing feature	example
best: boldface/normal	**while not** eof(input) **do**
underlined/not	while not eof(input) do
typeface change	while not *eof(input)* do

In contrast, the style suggested by Dr. Bishop is far too disruptive. This can be seen in her examples (8) and (9), and in the same fragment as used above:

WHILE NOT eof(input) DO

Indeed, the disruptive effect of the capitalization encourages the eye to mis-parse the fragment in the following way:

WHILE NOT / eof(input) / DO

FROM WHENCE DOES THE PRACTICE DERIVE?

It seems fairly obvious to me that the current phase of styles in publication Pascal derives from publication Algol 60. In that language there were no reserved words, and the distinguishing of basic symbols from identifiers was mandatory. However, in Pascal the situation is quite different and there is no requirement that the reserved words be specially singled out. So we end up with either the historical argument that Pascal reserved words should be distinguished because Algol basic symbols had to be, or with a circular argument Reserved words are distinguished because they are reserved words. . . . An alternative line of reasoning says that reserved words are distinguished because programmers do not define their meanings. Accepting this as an argument leads me to wonder why its proponents do not want to distinguish *integer* or *write* in the same way. After all, they too have pre-defined meanings. I draw two conclusions from this: (i) not all that we inherit from Algol 60 is good, and (ii) Pascal is not Algol and deserves its own special consideration.

In some of the styles reported by Dr. Bishop, the reserved words are italicized in contrast to identifiers in normal upright typefaces. I speculate that this curious practice may have arisen from the common printer's manuscript rules which regard underlining as a command to italicize. Some printers, especially those who set computing manuscripts frequently, have been trained to treat underlining as a request for a bold typeface.

LET US LOOK AT THE SAMPLES

Let me examine the sample pieces of program given in Dr. Bishop's paper for readability.

1. This is pleasing. However note that the switching between italics and an upright typeface is slightly distracting.
2. Irritating. PROGRAM, VAR and BEGIN are not too objectionable since they start major sections of the program, but DO and NOT do not need this emphasis.
3. Not essentially different from example 1.
4. The keyword distinctions are happily almost unnoticeable; the only jarring feature is the use of upper-case letters.
5. See example 4. However the underline facility here is definitely substandard: it runs right through the bottom edge of the characters.
6. Again see example 4. Here the 'overprinting' only succeeds in making the program look mucky.

7. Despite Dr. Bishop's remarks, example 7 is more legible than examples 2, 5 and 6. While lower-case is clearly preferable, if it is not available we have to take what is.
8. True underlining and good. See remarks on example 4.
9. Less readable than example 8. See remarks on example 2.

This analysis differs markedly from that of Dr. Bishop. My criteria are based on readability, hers on the success of distinguishing reserved words.

IS THERE A CASE FOR DISTINGUISHING ANY RESERVED WORDS?

I have argued that the set of reserved words in Pascal is sufficiently small that distinguishing them is not needed, and that such a practice degrades Pascal's readability for experienced users. To be fair, there is a sustainable case for distinguishing a few reserved words, deriving from a wish to be able to scan a program quickly for salient features. Visual cues can help in this process. Probably the main keywords in this class are *procedure*, *function*, *label*, *const*, *type*, *var* and an opening *begin*. A case might also be made for *end*, *else* and *until*.

While the argument is a valid one, it is not very strong. Other visual cues are much more effective, as can be seen by the effective use of indentation in Bishop's examples 8 and 9. Examples 6 and 7, by contrast, are poor.

COMMENTS

But the question of scanning program text raises a much more important point which I believe to be comparatively neglected: the treatment of comments. When I attempt to read a program as a string of syntactic symbols, comments are a disruptive influence. I want to ignore them, and concentrate on the next lexical token. On other occasions, the comments are of prime importance and I want to read them in preference to the program text.

I have always distinguished comments from program text, even way back in the days when I had to write in Fortran (circa 1964). There my technique was to link the leading C of a comment with its text by a string of '-'. Scanning down the usually sparsely filled left edge of the program gave me ample visual cueing after a bit of practice. Example:

```
C-----SCAN TABLE FOR A MATCH
      J=TMAX
      T(1)=VALUE
C-----WHILE CURRENT ELEMENT NOT=VALUE, COUNT DOWN
230   IF (T(J) .EQ. VALUE) GOTO 231
        J=J-1
        GOTO 230
C-----SINCE T(1) HAS THE VALUE WANTED, LOOP ALWAYS STOPS
C-----AND AT THIS POINT J IS THE INDEX OF THE MATCHED ELEMENT
231   IF (J .NE. 1) . . .
```

Now, in preparing Pascal programs for publication, I still prefer to distinguish comments more strongly than keywords, because they are for human consumption, and not for the machine. I do this by italicizing commentary; a good example is my recent paper on disciplined programming.[2] It is important to realize that preparing papers and programs for publication is not a task to be taken lightly; still less are textbooks with poor stylistics to be tolerated. In fact, in the paper referred to above, the printer was supplied with a full page of special instructions on typesetting, and to achieve the results we wanted we had to repaste up galley proofs to avoid splitting program text across page or column boundaries. An example will illustrate what I mean, so here is the Fortran fragment recast into Pascal:

```
{Scan downwards looking for a match}
index:=lastusedindex;
table [1] .key:=searchvalue;
while (table [index] .key < > searchvalue) do begin
  index:=index-1;
end;
{Since table [1] has the value wanted as its key, the loop always stops, even if only at this
sentinel. At this point index points to the record found.}
if (index < > 1) then begin . . .
```

IN TEXT

Of course, part of Dr. Bishop's letter addresses the presentation of program fragments in text, not simply the layout of programs themselves. I ignored this deliberately in the first part of the response so as to bring out the issues as separable. Programs in text must be treated in the same way as mathematical expressions: set off from the text by the visual cues of spacing and indentation. It is asking for trouble if the switch from discursive text to high formalism is not clearly marked. I have tried to do this in this paper. If this is adopted as a consistent practice, then there is no need to italicize programs as Dr. Bishop suggests. This is just as well, for it substantially increases the cost of typesetting or typing to provide a full set of italicized characters including digits, operators, etc.

For referring to program objects within text, Dr. Bishop remarks that it is seldom necessary to distinguish identifiers or reserved words. I agree. When it is, then we have the whole gamut of conventional techniques ready for use. Underlining reserved words is a possibility ingrained from long habit, but italicizing them as well as identifiers is probably better and more consistent. Compare:

The word *fly* is used to denote powered progress through the air, in contrast to *glide* which is unpowered.

The variable *a* is used to hold a lower bound, and *moving* indicates the direction of scan.

A PREFERRED STANDARD FOR PUBLICATION PASCAL

1. Pascal reserved words and identifiers should be typed or printed in the same typeface: all upright or all italics.
2. If lower-case letters are available, they should be preferred, especially for typewritten and typeset programs. Limited use of capitals for emphasis or readability is permitted.
3. In typewritten or typeset programs the distinguishing of reserved words from identifiers by underlining or the use of a bold typeface is acceptable, but not mandatory.
4. In typewritten or typeset programs the distinguishing of commentary from program tokens by the use of an italic typeface is acceptable, but not mandatory.
5. When appearing in prose text, programs and program fragments should be visually set off from the prose by line spacing and indentation so as to mark the switches between discursive text and formal text.
6. When referring to reserved words and identifiers in prose text, it is not always necessary to distinguish these from the prose. When it is, italicization is the preferred method for both reserved words and identifiers. When an italic typeface is not available, underlining may be substituted. Enclosing within quotes is not recommended.

IN CONCLUSION

The preferred standard set out above matches up well with reality. It can be applied to program listings (the output from compilers for example) as easily as to typeset material. We seem to be winning the battle for line printers and terminals with both cases of letters, but few devices can handle more than one type fount. The style suggested by Dr. Bishops' paper is therefore impractical for common use. Some of the other notions which have surfaced in textbooks are curious in the extreme; for example, the notion of 'overprinting' on a quality device with good registration is just silly.

I realize that this paper has avoided the issue of what place capitals do have in programming. This is deliberate, for it is intended to focus on one major issue: the distinguishing of reserved words, which I think makes programs even less readable than the all-upper-case style. The best I can do is to promise another polemic to follow this one, on good uses of capitals.

REFERENCES

1. J. M. Bishop, 'On Publication Pascal', *Software—Practice and Experience*, **9**, No. 9, 711–717 (1979).
2. C. Lakos and A. H. J. Sale, 'Is disciplined programming transferable, and is it insightful ?', *Australian Computer Journal*, **10** (3), 87–97 (1978).