Psychol. Belg.
1985, XXV-2, 169-174.

SHORT NOTE

HOW TO WRITE COMPOUND STATEMENTS IN STRUCTURED COMPUTER PROGRAMMING LANGUAGES?

André VANDIERENDONCK, Geert DE SOETE*
& Freddy VLAEMINCK
University of Ghent

An experiment is reported which demonstrates that novice computer programmers show a better comprehension of moderately complex computer programs composed according to an indenting scheme in which a new line is taken for each compound statement, as compared to programs in which the compound statement starts at the end of the line containing the statement. There was no interaction between the indenting scheme and the specific programming language used (RATFOR vs. PASCAL).

In the last decennium, research on the psychological factors involved in computer programming has immensely increased. It has become one of the main issues in the rapidly developing domain of software psychology (cf. Shneiderman, 1980).

Several research efforts have addressed topics introduced by the structured programming approach, which concentrates on principles that are thought to be essential in "good" programming. Although systematic empirical evidence for most of the principles is rather scarce, they are increasingly propagated in programming textbooks. More and more students in programming courses are thought to proceed top-down, to avoid goto's, to use a transparent style, to add adequate comments, to elucidate the program structure by means of appropriate indenting, to construct modular programs, to use meaningful variable names, etc.

Comparisons of programs structured according to these principles with equivalent programs that violate these principles have not always yielded unambiguous evidence in favor of the structured programming principles. Sheppard, Curtis, Milliman and Love (1979), for example, report that naturally structured programs are at least as easy to understand as strictly structured ones which are better comprehended than unstructured programs.

Geert De Soete is "Bevoegdverklaard Navorser" of the Belgian "Nationaal Fonds voor Wetenschappelijk Onderzoek".

Key constructs in most programming languages that allow structuring are the if-else clause and the compound statement. Complex branches must be made without use of goto's and labels. Therefore, each branch of the if-else statement must be filled with several statements that are considered as one block. In PASCAL these compound statements are marked with the keywords BEGIN and END. Anything comprised between two matching BEGIN-END pairs constitutes a compound statment. In RATFOR (Kernighan, 1975) and C (Kernighan & Ritchie, 1978), compound statements are formed by grouping statements between a pair of matching braces ({and}). Although RATFOR programs are preprocessed into FORTRAN 66 prior to compilation, RATFOR allows for C-like control structures.

Without use of a proper external structure a compound statement may be hard to detect within the body of a program. Therefore, the use of an indenting scheme can be very useful to visualize the program structure. However, advices about indenting schemes are often contradictory. Some authors suugest to begin a compound statement on a new indented line. After the opening marker, a new line is taken and all statements within the block are indented with respect to the marker. The closing marker is also on a separate line aligned with the opening marker (cf. Table 1, PASCAL (i)). This is a scheme often used in PASCAL textbooks (e.g., Ledgard, Nagin & Hueras, 1979).

Tab. 1. — Indenting schemes in PASCAL and RATFOR

	(i)	(ii)

PASCAL	if a > b then
	begin
	x : = a+b;
	y : = a−b
	end;
		if a > b then begin
		x : = a+b;
		y : = a−b
		end;

RATFOR	(i)
	if (a > b)
	{
	x = a+b
	y = a−b
	}
		(ii)
		if (a > b) {
		x = a+b
		y = a−b
		}

In their description of software tools in PASCAL and RATFOR, Kernighan and Plauger (respectively 1981, 1976) propose a radically different indenting scheme (cf. Table 1, (ii)). The compound statement is opened on the same line as the condition to which it applies; on the next few lines the body of the compound statement is indented, and the

closing marker is placed on a separate line. In a PASCAL application the end is aligned with the line starting the compound statement, while in a RATFOR application, the right brace is indented to the same level as the statements within the compound block.

For computer programmers, textbook writers and teachers, it is interesting to know which scheme most enhances program understanding. Previous research by De Soete, Vandierendonck and Vlaeminck (1985) has shown that (a) comprehension of small programs consisting of if-else clauses, compound statements and one-word executable statements is fostered by the use of an indenting schema, and (b) that indenting interacts with the kind of program code syntax (PASCALlike vs. RATFORlike). A positive effect of indenting on the comprehension of conditional clauses was observed only with the PASCALlike syntax.

In order to increase our understanding of this interaction and in order to answer the very practically oriented question stated above an experiment was performed in which the effects were studied of the two indenting schemes in both PASCALlike and RATFORlike contexts.

METHOD

Subjects and design

Thirty-six students (17-18 years old) from a Belgian secondary school volunteered to participate in this experiment, which lasted for about 1 hr. They were radomly assigned in equal numbers to the cells of a 2 (RATFOR vs. PASCAL) × 2 (indenting scheme) factorial design. The subjects did not have any prior programming experience.

Materials and procedure

Table 2 contains two examples of "programs". The subjects were told that the programs are collections of commands used to monitor a robot in a vegetable canning factory. They were explained the syntax of the if-else clause and the compound statement. When all instructions were clearly understood, two programs each followed by a set of 10 questions about the functioning of the programs were presented. (The Appendix contains an example of 10 questions corresponding to the program displayed in Table 2).

All subjects were tested simultaneously in one group. After a short explanation by the experimenter, they were asked to study the instructions. When everything was understood, the experiment started. Subjects

Tab. 2. — Sample problem in PASCAL and RATFOR with both indenting schemes

```
IF (round) {                    IF round THEN
    IF (juicy)                      BEGIN
        IF (green)                      IF juicy THEN
            roast                           IF green THEN
        ELSE                                    roast
            cut                             ELSE
        }                                       cut
ELSE                                    END
    peel                            ELSE
IF (big)                                peel
    IF (hard) {                     IF big THEN
        IF (red)                        IF hard THEN
            bake                            BEGIN
        }                                       IF red THEN
    ELSE                                            bake
        refuse                              END
IF (leafy) {                            ELSE
    IF (green)                              refuse
        roast                       IF leafy THEN
    IF (red)                        BEGIN
        cook                            IF green THEN
    }                                       roast
                                        IF red THEN
                                            cook
                                    END
```

were encouraged to work fast and accurate. A debriefing followed immediately after the session.

RESULTS AND DISCUSSION

Correct responses were scored 1; errors or ommissions received a zero score. These scores were subjected to a 2 × 2 fixed effects ANOVA. The difference between PASCALlike and RATFORlike programs was not significant, $F(1, 32) = 0.09$, $MS_e = 14.18$; respective means: 10.61 and 10.22. However, there was a clear effect of the indenting scheme, $F(1, 32) = 8.28, p < .01$. The scheme with the compound statement starting on a new line was superior to the scheme proposed by Kernighan & Plauger (1976, 1981). The respective means amounted to 12.22 and 8.61. The two factors did not interact significantly ($F(1, 32) = 2.13, p > .10$), although the cell means in table 3 suggest that the superiority of indenting scheme (i) is more pronounced with PASCALlike syntax than with RATFORlike syntax.

Tab. 3. — Mean performance as a function of indenting scheme and syntax

Syntax	PASCAL	RATFOR
Scheme (i)	13.3	11.1
Scheme (ii)	7.9	9.3

These results suggest that, at least for novices, a PASCALlike code is not easier to comprehend than a RATFORlike code. This may be considered a plausible outcome since both codes are highly structured and differ only in a few syntactical details. The RATFOR if-else puts the condition between parentheses, whereas the PASCAL if-else requires a THEN keyword between the condition and the branches.

The main finding in the present study concerns the superiority at least for novices of an indenting scheme that treats compound statements as separate units starting on a new line. It is not clear whether experts would benefit in the same way from such an indenting scheme, that certainly suffers from a number of drawbacks. Firstly , for every compound statement, the code is incremented by one line when using indenting scheme (i) in comparison to scheme (ii). Secondly and more importantly, scheme (i) requires two indentations, one for the begin and end markers and one for the body of the statement. Consequently, nested compound statements can become so deeply indented that any perceptual benefits are lost.

The present data, therefore, suggest a partial answer to the question of how to indent. If the users are novices and the compound statements are not very deeply nested, then a scheme starting a new line for each compound statement should be preferred. Further research should shed more light on the practical significance of the indenting schemes for experts and for complex programs. Also, alternative indenting disciplines as the semantically based formatting scheme proposed by Bailes and Salvadori (1984) should be empirically evaluated.

REFERENCES

Bailes, P. A., & Salvadori, A. (1984). A semantically-based formatting discipline for Pascal. *Software-practice and Experience, 14*, 235-251.

De Soete, G., Vandierendonck, A., & Vlaeminck, F. (1985). *Effects of indenting, commenting, and syntax on the understanding of conditional clauses in computer programing*. Unpublished manuscript.

Kernighan, B. W. (1975). RATFOR — A preprocessor for ratrional Fortran. *Software — Practice and Experience, 5*, 395-406.

Kernighan, B.W., & Plauger, P.J. (1976). *Software tools*. Reading, MA: Addison-Wesley.

Kernighan, B.W., & Plauger, P.J. (1981). *Software tools in Pascal*. Reading, MA: Addison-Wesley.

Kernighan, B.W., & Ritchie, D.M. (1978). *The C programming language*. Englewood-Cliffs, NJ: Prentice-Hall.

Ledgard, H.F., Nagin, P.A., & Hueras, J.F. (1979). *Pascal with style. Programming proverbs*. Rochelle Park, NJ: Hayden.

Sheppard, S.B., Curtis, B., Milliman, P., & Love, T. (1979). Modern coding practices and programmer performance. *Computer, 12*, 41-49.

Shneiderman, B. (1980). *Software psychology*. Cambridge, Mass.: Winthrop.

APPENDIX

Questions corresponding to the sample program shown in Table 2

1) What happens to big round objects when they are juicy, hard and red?
2) What happens to big leafy objects that are not hard?
3) Which features do objects have that are to be peeled and roasted? (it should not be refused).
4) What happens to green objects when they are round, juicy and big but not leafy?
5) What happens to green objects when they are round, juicy, big, hard and leafy? (Attention: give a complete answer).
6) What happens to green objects when they are not big and not hard?
7) What happens to green objects when they are big, hard, leafy and not round?
8) Which properties should an object possess if it is to be baked?
9) Which properties should an object possess if it is to be cut? (Other actions on the object are irrelevant).
10) What happens to a big yellow object which is hard, leafy and not round?

André Vandierendonck
Department of Psychology
University of Ghent
H. Dunantlaan 2
9000 Gent

Received May 1985