This forum is offered for readers who want to express their opinion on any aspect of information processing. Your contributions are invited.

# the forum

## Structured BASIC is Alive and Well in Denmark

In his excellent article, "Structured Programming in BASIC" (June, p. 149), Peter B. Worland states: "Ideally the structuring should be built into the compiler/interpreter." Such an interpreter has been designed. It was implemented on a Nova 1200 in 1975 here at the Computer Science Dept. of the Government Teachers' College, Tonder, Denmark.

The interpreter is now used at most schools and institutions in Denmark having RC 7000 minicomputer systems. (The Nova 1200 central processor unit is a component in the RC 3600 and RC 7000 minicomputer systems, manufactured by A/S Regnecentralen in Denmark.)

And by now we have gained considerable experience with our Structured BASIC, which we call COMAL (COMmon Algorithmic Language). The following are it's most important features:

*1. Variable names*
Variable names may contain as many as eight characters.

*2. Assignments*
LET statements may contain more than one assignment, for example:

LET COMCODE=COMCODE+1; P2=COMCODE*3; P1=P2-2

*3. Control structures*
In addition to the control structures available in BASIC, COMAL provides:

*a. IF p THEN . . . ELSE . . . ENDIF*
*b. IF p THEN . . . ENDIF*
*c. REPEAT . . . UNTIL p*
*d. WHILE p DO . . . ENDWHILE*

Thus there are three loop structures in COMAL: FOR . . . NEXT, REPEAT . . . UNTIL, and WHILE . . . ENDWHILE. And there may be seven of each type nested in one another regardless of the order, and seven IF branch constructions too. In all cases, *p* is a Boolean expression, and the program text is indented for easier reading.

Still another control structure is:

*e. CASE expr OF . . WHEN <list$_1$> . . WHEN <list$_n$> . . . ENCASE*

The *expr* may be an integer expression, Boolean True or False, or a string variable; each *list* may be a list of integers, Boolean expressions, or strings. On processing, each list is examined to see if it fits the present case, and a default case may be inserted for the times when no list fits. In COMAL, the number of WHEN's used for any given CASE is not limited, and CASE . . . ENDCASE construction, may be nested to any depth.

*4. Boolean expressions*
Full Boolean algebra with the Boolean operators AND, OR, and NOT is available. COMAL also has inherited the pseudo-Boolean expressions from Extended BASIC: a numerical expression is in proper context considered false, if it has a value of 0, and true in all other cases. A numeric variable may be assigned the value of a Boolean expression, 1 if the expression is true, and 0 if the expression is false. Also, Boolean expressions may be included as parts of algebraic expressions with the same evaluation.

*5. Subroutines*
If a program is initiated with the statement *PROC* <name> where *<name>* is a string formatted as a variable name, and is terminated with the statement *ENDPROC*, this program may be called as a subroutine by another program using the statement *EXEC <name>*. When the subroutine has been executed, control is passed to the statement following the EXEC statement that called the subroutine.

If you think that EXEC is nothing but GOSUB, well, try it!

Furthermore, the program text between the PROC and ENDPROC statements is indented in the program listing. This makes it easier to identify the subroutine.

A subroutine may call another subroutine, but there may not be more than seven nested subroutines.

*6. Comments*
The keywords ELSE, ENDIF, REPEAT, ENDWHILE, ENDPROC, and END have the same status as REM with regard to comments. In other words, an explanatory comment may be inserted after any of these words.

### Small enough and plenty fast too

By now you may think that a Structured BASIC interpreter is a huge affair compared to the Extended BASIC interpreter (which is still a proper subsystem of ours). But this is not so. In fact, we added only 12% to 15% more code to what was already there. Size is therefore no excuse for continuing to use an outdated, inadequate language like Dartmouth College BASIC, simply "because it *was* there," as Dr. Worland aptly remarks.

The first version of COMAL was implemented here at Tonder in 1975, on a Nova 1200 with 24K words of core

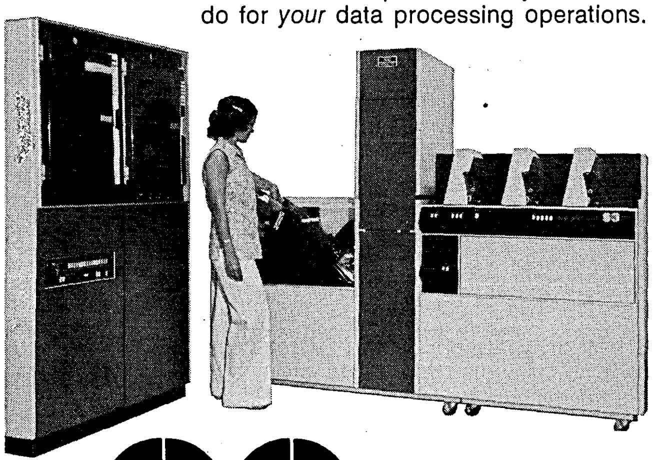
**IN THE WORDS OF A CUSTOMER . . .**

**Four ways a Scan-Optics OCR system helps the profitability of O.M. Scott & Sons**

*Dennis McMullen, manager of information processing*

"*One,* it has reduced data entry costs by over $9,000 a month . . . *Two,* it has facilitated decentralization of data entry operations, making the capability available to the order processing, purchasing, accounts payable, accounts receivable, manufacturing, marketing and subscription departments . . . *Three,* it has reduced the number of errors and improved the accuracy of information entered . . . *Four,* it has made data entry available to those who have no specialized data entry devices except typewriters and pencils."

Find out what Scan-Optics OCR Systems can do for *your* data processing operations.



SCAN-OPTICS, INC.
22 Prestige Park
East Hartford, Ct. 06108
203/289-6001

CIRCLE 132 ON READER CARD

## the forum

and paper tape peripherals. The whole thing was done on paper tape—miles of it. At that time, the smallest version took 12K words of core and extended versions took 14K. (Some of these versions are still running at some high schools in Denmark.)

We've been running our current version for over a year, this time under RDOS on a 32K Nova with a Diablo moving head, 2.4MB disc. This same version is used by some large school systems and by the Univ. of Aarhus. COMAL *and* the operating system take up 22K of the 32K memory.

There are three of the very latest versions: (1) a mini-COMAL meant for small standalone machines with at least 16K; (2) a midi-COMAL for machines with at least 24K and dual floppy discs; and (3) the maxi-COMAL for cpu's with at least 32K and either dual floppies or a hard disc. The mini version uses 12K words, the midi 15K, and the maxi 18K. (The maxi version, by the way, handles integer, real, and Boolean data types and strings. It also handles arrays of all the mentioned types and 14-digit arithmetic.)

We plan to produce a version for the Zilog 80 microcomputer too, but are having difficulties in finding a good BASIC to build it on.

The original Extended BASIC was reasonably fast compared to other BASIC versions I've seen. COMAL is faster, sometimes up to 40% faster than equivalent BASIC programs, because the structured algorithms may add substantially to efficiency.

Here is an example: If one implements a branching by means of GOTO <line number>, the original interpreter will start scanning the line numbers from the beginning of the program until it finds the right one.

In COMAL, the interpreter "knows" that it has to look forward for an ENDIF or an ELSE, depending on the value of the Boolean expression in the control statement IF . . . THEN. Similarly with REPEAT . . . UNTIL and WHILE . . . ENDWHILE. When you work with structured programs in a way you work with "hidden labels."

By throwing away the GOTO completely, it would be possible to make a very fast COMAL, because we might then "semicompile" the source text, thereby making jumps and subroutine calls as fast as just looking for "the next line."

The impact of Structured BASIC in the schools has been significant. Not only are students writing better, more valid, and more readable programs—that was to be expected—but they also are being encouraged to write more ambitious programs due to the relative ease with which one can grasp a well-structured program. Perhaps I should add that the long variable names have proved to be of great importance in making programs self-explanatory, and some of my colleagues even say that this facility is the last one they would be without.

Classical BASIC may be of use in laboratories for short numerical problems, or in the newest hand-held calculators, but it has long ceased being *the* language for schools, and the fact that so many teachers still think it can only be due to their ignorance of more modern ideas such as those found in PL/1, PASCAL, or SIMULA.

One last remark: our COMAL interpreter has been taken over by the Danish computer manufacturer A/S Regnecentralen for further development and maintenance, and Structured BASIC is now spreading throughout Europe with the RC 3600 and RC 7000 minicomputer systems.

—**Borge R. Christensen**

Mr. Christensen is the computer center director and a lecturer in math and informatics at the Government Teachers College, Tonder, Denmark. He is also the co-developer of COMAL.