TEACHING GOOD PROGRAMMING TECHNIQUES

J. L. Lowther and Z. C. Motteler
Michigan Technological University

A goal of any programming class should be to design and write readable, understandable, and correct programs. Too often, a beginning programming class will cover language features and generate examples illustrating language features, presenting very little, if anything, concerning proper programming style and methodology. At Michigan Technological University, such a situation exists. CS 110, Basic FORTRAN, is a multisection two quarter hour course taught by instructors from a wide variety of backgrounds. Because of the time constraints of the course, very little beyond the basic features fo FORTRAN may be taught. Students who emerge from this course can typically produce "correct" programs, in the sense that these programs do what they are supposed to in a sort of minimal sense, but poor programming practices are rife. Programs are riddled with hopelessly complicated control structures, output is poorly formatted and there is often no heading to tell what it is, programs have either no comments at all or so many that it is difficult to find the code, and so forth.

To remedy this situation, CS111, Advanced Programming Techniques, emphasizes good programming habits and encourages the design and writing of readable, understandable, and correct programs. In addition to teaching students structured and topdown programming techniques, proper documentation, etc., a major goal of the course is to inculcate into them a professional attitude twoard their programming. No program ever receives an A if, the instructor's judgment, a very particular (and knowledgeable) employer would send it back for further work. Programs must be complete, comprehensive, reusable, and easy to alter.

The portion of the course which teaches structured programming techniques is particularly interesting. The students are introduced to D charts and D structures, following Mills [1], i.e., composition, alternation (if-then-else structures), and iteration (do while structures). These are then extended to what Ledgard and Marcotty [2] call D' structures (if-then, repeat-until, and case statements). The students learn how to recognize non-well-structured programs and flowcharts and various techniques to convert them to properly structured programs (as described for instance by Yourdon [2], pp. 153-168). Finally, they are encouraged--even required--to produce well-structured programs in fulfillment of the course.

Although Algol, PL/1, and other Languages available on Michigan Tech's Univac 1110 have many of the structured commands available, these languages are not as fully implemented and reliable as FORTRAN. As a result, faculty members have designed two FORTRAN preprocessors [4, 5], the first of which allows the D structures and an INCLUDE statement, the second of which has a rich repertoire of such extensions as REPEAT(...)TIMES blocks, REPEAT...FOREVER, BEGIN blocks which may be EXECUTEd as internal subprograms, and EXIT, CYCLE, and RESTART statements. Students are instructed in the use of these preprocessors and are strongly urged to use them to pursue GOTOless (and EXITless, etc.) programming.

Accompanying the material on programming style are discussions on file usage, round-off error, plotting, trace features of a compiler, reading dumps, modular programming, and other topics relevant to Computer Science. The excellent little paperback by Kernighan and Plauger [6], is required reading for the course and many problems from the text are assigned. These provoke considerable thought and debate in class, frequently directing the instructor's lecture toward advanced or subtle points in the language which are not well understood by students.

As a result of CS111, Computer Science freshmen and many students from other fields as well become aware of the importance of good programming techniques and consistently produce "beautiful" programs throughout their career in Computer Science.

References

1. Mills, H. D., The new math of computer programming, Comm. ACM 18 (1975) pp. 43-48.

2. Ledgard, H. F., and M. Marcotty, A geology of control structures, Comm. ACM 18 (1975), pp. 629-639.

3. Yourdon, E., Techniques of Program Structure and Design, Prentice-Hall, Englewood Cliffs, 1975.

4. Lowther, J. L., "MTUFP, Michigan Technological University FORTRAN Preprocessor," Structured FORTRAN Preprocessor Survey, UCID - 3793, University of California-Berkeley, 1975, edited by Reifer and Meissner.

5. Motteler, Z. C., "ZCMP Preprocessor," submitted for publication.

6. Kernighan, B. W., and P. J. Plauger, The Elements of Programming Style, McGraw-Hill, New York, 1974.