Session I
Structured Programming: Concepts and Definitions

Overview
John Naughton
IBM Federal Systems Division

Session I addressed the concepts and definitions of structured programming (SP).
Leadoff speaker was Lynn Maxson of IBM, who reviewed the terminology and definitions applicable to theory, practice, and related techniques. He approached these ideas from a systems perspective, capitalizing on the work of Larry Constantine in structured design to support his presentation.

According to Maxson, “Structured programming activities fall into three categories: methods, means, and management. Within methods are structured design, structured programming, and structured walkthroughs. Both design and programming use top-down development for the ordering of their developing process. The means employed to record the structured methods are the development support library, HIPO diagrams, and structure charts. Finally, the management used to enforce both the means and methods is the chief programmer team.”

The participants attending the workshop indicated an overwhelming interest in design, data structuring, and programming languages. The session was constructed to address this interest. McGowan and Reifer concentrated their presentations on the design and data structuring area.
Stewart and Horowitz discussed SP concepts as they relate to programming languages in general, and specifically to FORTRAN.

During the presentation on languages, which contained information on approximately ten FORTRAN precompilers, there was much discussion and interaction on the question of standardizing SP figures and syntax. The theoretical work of Bohm and Jacopini prescribed a sufficient set of figures for constructing proper programs. Many individuals are now working on developing an optimum set for contemporary environments and applications. This activity is leading to a plethora of language approaches for implementing SP. The point was made that this activity should be controlled throughout the industry. One approach suggested was to concur on a basic set of figures—including syntax and semantics—and use these as an industry standard for production programming. This would help reduce the problem of coping with many SP language dialects.

The second part of the approach was to continue the work in researching control structures with the objective of optimizing them. When it is determined that the theoretical work (in whole or part) could be introduced into programming practice in any constructive way—then do it. The question of who would or could control SP language development was not answered.

Dr. Clement McGowan of Brown University discussed top-down design and implementation. Central to this approach is the notion of a code segment, which is a concise realization of some function. System design is developed top-down and expressed in pseudo-code segments, which are short structured programs for abstract machines. The design process is continued until all the principal data representation decisions can be intelligently made; then top-down coding commences. McGowan, who touched on macro includes, program stubs, and the joint use of master and development libraries for version control during system implementation, stressed the advantages of top-down implementation over the more traditional bottom-up strategy (e.g., in testing and integration).

Mr. Don Reifer of the Aerospace Corporation spoke on SP as a support tool. In order for SP to realize its full potential as a scientific discipline, said Reifer, it must be extended to provide timely support to individuals making critical requirements and design decisions. Presently, this is not accomplished because the discipline employed is not applied to document user needs, specify software requirements, or structure data or data flow. According to Reifer, as a support tool for structured design, SP includes notational conventions, restrictions on flow of control, and specified access to data.

Dr. Seldon Stewart discussed STAPLE, a structured language based on FORTRAN, designed to investigate the special case of a completely nested block-structure syntax in which flow of control is constrained to the tree defined by the block structure. The scope of variables was not determined by the block structure. For top-down design and code readability, methods for invoking blocks of code by name are extremely useful, including the independently compilable subprogram, procedures which satisfy the copy rule, and macro insertion of a code block.

Carefully skirting the problem of settling on a formal definition of SP, Dr. Ellis Horowitz of USC emphasized instead that it should not be taken as merely the elimination of GO-TO's nor as the writing of programs using only the three basic constructs. The major share of Horowitz's talk was a summary of seven structured FORTRAN pre-processors. Features included IF-THEN-ELSE, WHILE, UNTIL, LOOP, EXIT, FOR, and CASE statements. Among the advantages of these systems were (1) that they provide these useful constructs, (2) they are economical to run, (3) the resulting code is efficient, and (4) they allow all legal FORTRAN, making programmer conversion easier. However, people are reluctant to change, deliverables can cause a problem, and code increases can be 10 to 15%. The question of standardization of these features provoked a lively discussion.

Finally, the problem was addressed as to what minimum set of logical constructs should be adopted for SP. The argument centered around the two-predicate figure that is used to represent a table search algorithm, and also whether a control mechanism should be permitted which facilitates an alternative exit criterion from a looping operation. Finally, a proposed syntax was presented to permit SP in ANS FORTRAN, covering the conditional statement, both looping control structures (WHILE and UNTIL), and a case statement.

Some of the issues mentioned above are dealt with in a more thorough fashion in the two papers that follow, both of which were selected from Session I.

John J. Naughton is currently manager of an IBM effort to transfer structured programming technology to Rome Air Development Center for wide-scale dissemination and to investigate and develop related structured programming techniques which have the potential to improve the development and management of software. Formerly, he was the responsible manager in developing the guidelines and tools necessary for the enhancement of programming support libraries, structured programming, top-down programming, and chief programmer teams. Prior to his work in structured programming, he managed and worked in a number of command and control and intelligence data base/data communications systems with IBM.

Mr. Naughton received his BS in physics from St. Joseph's College in 1957.