The programming practices in geology

Tom Svane Petersen and Gyrite Brandt

Petersen, T. S. and Brandt, G.: The programming practices in geology. *Danm. geol. Unders., Årbog 1978*, pp. 173-180. København 1979.

The purpose of this paper is to show a part of the essence of efficient geological programming. In this context, programming assumes the meaning of communication between the geologist and the computer. The programming process is subdivided into five parts, and a discussion of each of these parts is presented, special emphasis having been made on the step concerning the choice of programming languages.

Tom Svane Petersen. Geological Survey of Denmark, Thoravej 31, DK-2400 Copenhagen NV. Gyrite Brandt, Geological Centralinstitute, Østervoldgade 10, DK-1350 Copenhagen K

Due to widespread availability of computers and the accompanying facilities, these are now being used in nearly all sciences as tools for solving a large variety of problems. Geology has shared a part of this revolution. Geologists, however, have been slower in realizing and using the advantages of the computer than other scientists in fields. The main reason for this is related to the fact that geology is mainly a qualitative and descriptive science. In recent years large quantities of articles, dealing with computers, have been published. This is due to, among other things, the quantity of data that can be recived from the automatic analysis instruments. Many people can utilize programs for their work without having to know any of the basic facts about a computer. It is a necessity for the geologist to learn a programming language, and the number of these at the present stage is multifarious.

If the only person using the program is the programmer himself, then it is possible that adequate instructions may be kept in a loosely organised form in the programmer's own mind. It is most likely, however, that written notes will have to be made even for the programmer's personal use. If another person is to use the program, these instructions will have to be communicated to him, and the programmer will thus have to prepare a set of well-organised, detailed instructions in writing.

The subdivision of the programming process is a subjective matter, and in any attempt to define the different tasks it should be realized that these individual

steps are interdependant. However, most experienced programmers will probably agree that the following steps make up a reasonable description of the major tasks involved in writing a program:

1) Problem definition.
2) Selection of algorithms and programming language.
3) Specification of program logic and structure.
4) Testing of the program.
5) Documentation.

These steps are certainly not independent. Appropriate written records should be made during the first steps; these will later make up a significant portion of the documentation as a whole.

Thus documentation is not a separate step but an intergral part of the programming. In the following sections we will discuss these steps, hoping to indicate some good geological programming methods and to show which programming languages are useful in geology.

Problem definition

The first obvious step in solving a problem in geology is to determine, in exact terms, what the actual problem is. The same is the case when a geologist wants to solve a problem with the aid of a computer. Though this may seem obvious, many geologists and other programmers will frequently start programming before they have an absolute understanding of the problem. There are several reasons why this is so, the first reasons usually being that the person who is faced with the problem does not thoroughly understand it. When the geologist and the programmer is the same person, he may be able to discipline himself so as to make an effort to define the problem properly before proceeding. In many cases some preliminary computation will be an aid in defining the problem, and this should be performed. When the geologist is not the programmer, he must be sure that the problem is welldefined, or else any cooperation with the programmer will be extremely difficult.

When the geologist is not the programmer, he must always remember that the programmer is confined to working on his own understanding of what he has been told. When considering the difficulty of communicating even the simplest ideas, it is no wonder that the programmer may misunderstand or misinterpret even the most rigid definition of the problem that the geologist has set up. The first step towards an understanding of the geological problem to be solved, is a personal meeting between the programmer and the geologist. Here the main difficulties should be discussed and eliminated before the working out of detailed specifications is begun.

The role of the programmer in this process should not be underestimated. The programmer is certain to have a greater appreciation of the capabilities and limitations of the computer than the geologist. It should also be noted that the geologist has probably made his definition based on co-operation with another geologist, and the programmer. The programmer will not have the background knowledge to enable him to interprete the definition. In the initial to specify the problem, there may be certain difficulties in making the definition ridgid enough. This may be because the problem may not be completely understood until more information is available in some cases, and this is a very common situation in geology! In other cases, it may be so that the solution to the orginal problem suggests an extension or modification that may be of interest. For these reasons it is necessary to maintain enough flexibility in the specifications and subsequently in the design of the program, so as to allow changes to be made easily. It is obvious that the problem definition and program specifications will be important parts of the final documentation, and therefore these parts should be made clear, by the geologist right from the beginning.

Selection of algorithms and programming language

Once the problem has been identified, the next step is to determine the methods to be used in solving it. In some cases these may have been determined by the specifications, in other they may depend heavily upon the language and equipment used. In geology the question of using a computer or desk calculater for programming procedure, depends on at least one of the following four criteria, (Harbaugh & Merriam 1968):

1) Volume of data to be processed.
2) Number of mathematical manipulations.
3) The volume of data to be retrieved.
4) Availability of the program.

In practice, the selection of algorithms does not involve many problems, as the solving of geological problems in general is not a complex affair, and it is also possible to consult the book by Knuth (1973).

When the necessary algorithms have been located, the next step for the geologist should be to determine whether related programs are already available, but obviously the amount of effort made in locating a program should be compared with the amount of effort required to write the program. In the event that the necessary programs are not available, there are perhaps programs which will accomplish the same task through a different, but perhaps acceptable program. Also, perhaps, an existing program could be modified to accomplish the task at hand. All of these possibilities should be investigated. Though no existing program seems to

solve the entire problem, there may be an existing one that is applicable to a portion of it.

In the early days of computing, programs were written in numerics, which then were translated to numeric equivalents through a simple translation program. Development of the problem orientated languages began in the late 1950's and this development has progressed rapidly since, the result being that today's programmer is confronted by a nearly bewildering variety of tools. The most basic of these tools is microprogramming in wich the programmer has control of the gates, flipflows, and registers within the central processor, the minutest details concering each task having to be specified. The so-called machine language, or assembly language, is still the most common tool available to most programmers. The geologist should realize that the characteristics of the assembly languages are strongly dependent upon the characteristics of the computer itself, and the programs written in assembly language are generally unsuitable for any machine other than the one for which they have been written.

Beyond the assembly language we encounter a host of problem orientated languages, and these are the most useful for solving geological problems. These range from business data processing languages such ad Cobol, through mathematical-type languages such as Fortran and Algol, multiple-purpose languages such ad PL/1, list processing languages such as Lisp, pattern-driven languages such as Snobol to array-processing languages such as APL.

A difficult problem for the geologist is the selection of a programming language. Here, the issue is not only how flexible or sophisticated a programming language is, but also how widely used it is plays a great part when one is to make one's choice. For a specific survey or university this last problem is not of great importance, but if the geologist is interested in making the program available to other geologists, this is an essential problem. If one looks at the geological publications on computer programs issued within the last ten years, one will see that more than 90 percent of the programs have been written in fortran. Also, all computers have a Fortran compiler, including the mini-computers wich are in rapid development. So in all probability most geological institutions will have a computer in the near future. In view of the above, and at the same time considering that Fortran is a very easy language to learn, it seems that Fortran is the relevant language for geological programming.

Program logic and structure

The next step is for the geologist to define the logic and to determine the structure of the program. This phase is the most important as the coding and ease of reading of the program depends on it.

Some geologists may feel that parts in this phase are important enough to be regarded as steps in their own right. We have grouped them together so as to emphasise their interdependance. We have divided this step into three subsections:

1) Modularization.
2) Logic via flowchart.
3) Structure of the individual module.

The process of dividing the program into basic building blocks is known as modularization. While the idea is simple, its implementation is generally not. The major difficulties involved are the determination of a suitable set of subtasks for each task, and the decisions related to the defining of size of these. For a geologist, modularization may seem to be unnecessary but this is not so as it makes it easier for the programmer and the reader of the program to read it.

One common method of expressing the logic and structure of a program is using a flowchart, and this visual aid can also be used by the geologist for solving problems that are not even to be programmed! A flowchart is a schematic visual description of the logic of a program module. Normally, each module should be flowcharted unless its function is so simple that the logic is clear. Flowcharts are primarily intended as tools for human communication, and these must therefore not be too general or too detailed. If it is too general, it will be of no value, e.g. a chart that shows in- and output boxes is not very informative but technically correct. On the other hand a flowchart must not be too specific as it will be hard to read. It is difficult to say what a flowchart should show. A flowchart should reveal, to the reader, the function of the program and the logic of the program! because of this it is impossible to say how many coded lines each box should represent.

Modularization of the program is very important. The clarity of the program as a whole depends heavily on the structure of each individual module. With regard to the flowchart, complexity may be thought of as being by the number of nonsequential paths through the chart, and this is an exponential function of the number of decisions.

A program is easy to read if it is written sequentially. On the other hand if the resulting items in the output list appear in random order with respect to the display of these, then it will be extremely difficult to read.

Input and output data format may be determined as a part of the program specifications. Often these formats are not especially important with regard to the overall structure of the program however.

Input data format should be designed for maximum user convenience. Every effort should be made to determine what format and sequence would be natural from the geologists point of view, and any necessary reordering or reorganization

should be an integral part of the program. If data manipulation is substantial, this will, of course, be included in the structuring and design of the program.

Output format is generally another matter. The design of the program may be very specific in terms of what information is to be written out, and when this is to be done. However, the specifications are addressed for identification of the output end.

The same standard of documentation should apply to the output as to rest of the program. The output documentation should be brief, and one should be able to read it separately. In particular, the output should begin with an introduction which identifies which program is being used, perhaps even stating the purpose of the program and the programmer's name. This should be followed by the identification of the problem, including any appropriate narrative description which includes a summary of the input data used in the run. The output data format should be so designed that it ensures maximum readability. Every data item should be positively identified, (including the input data value). This identification should be sufficient to allow for easy interpretation of the output by the geologist, but it should not clutter up the page. If the output information is printed in tabular form rows and columns should be identified and the columns aligned. Logical group of data should be separated by blank lines or page ejects. Identification, spacing and other good formatting practices should be used as a matter of course. The aim should be to produce an output that may be photographed and published without further editing, in the event that publication should become desirable.

Testing of the program

The first step to be taken by the geologist in this check-out procedure is to review the problem statement, the algorithms, data structures and the flowchart. By doing this, one should be able to reaffirm that the problem has been completely understood, and that the technique used for solving it will actually do so. A good idea is to get a geologist who is unfamiliar with the problem to examine the program and the program method. Whilst checking the program the geologist should make some calculations. The next step to be taken is the coding of the program. When the program has been coded, a print-out of it should be checked. The next hurdle is the compilation. Initially this should be done using the best diagnostic compiler available. A good diagnostic compiler is able to discover errors such as misspelt variable names, undefined variables etc. After a successful compilation an actual testing of the program can begin. The key to successful testing is that of good test data. The geologist should realize that the design of good test data is nearly as important as the design of the program itself. The test data should be designed so that it will test all coding, and the modules should be tested in as

many combinations as is practicable. It is often virtually impossibly to test all the paths through the program but the most common ones should be tested, and all the segments of the code executed.

Documentation

There are two classes of documentation of a program: the technical documentation, and the operational documentation. The operational documentation should include the following items: 1) The complete problem statement. 2) Algorithms and data structure. 3) Description of the logic and the structure of the program (including flowchart). 4) A clear, sequentially numbered, commented program listing. 5) A definite statement of testing and verification procedures. This statement should include a listing of test data, a description of which program sections were tested by each data group, and a description of the verification procedure. 6) Operating instructions, including program deck structure, data deck structures, card formats, output format, error messages and other abnormal conditions.

Technical documentation is nearly the same as operational documentation. The difference is that most geologists are not particularly interested in the mechanics of the program and therefore do not want bulky documentation. Generally the problem statement may be condensed, and algorithms and data structures may be presented in only just sufficient details to enable the user to understand what the program does, and why it does it.

Conclusion: If one makes programs that are to solve geological problems, one must take note of the following points: Problem definition (a clear communication between programmer and geologist is needed here); a selection of algorithms; a specification of program logic and structure; a testing of the program and documentation.

One must make certain that one uses a widely used and flexible programming language, and also make sure that one's written description is clear and easily understood. These two points will make it possible for others to use the program without unnecessary complications.

Last but not least it is important to stress the need of programming for geology in general, and for the geologists themselves, in order that geology may remain a modern and up-to-date science.

Dansk sammendrag

Fortran fremhæves som det bedste sprog til løsning af geologiske problemstillinger.

Der foreslås og gennemgås en metodik, som geologer kan følge når de skal løse programmeringsopgaver.

References

Harbaugh, J.W. and Merriam, D.F. 1968: Computer applications in stratigraphic analysis. John Wiley, New York.

Knuth, D.E. 1968-1973: The art of computer programming. 1, 2, 3. Addison-Wesley Publishing Company, New York.