Instructions for the final job

Think carefully what the program has to do, and specify that precisely.

If possible, implement the program as a module that can be linked to the user program.

Use efficient methods and avoid unnecessary calculations.

Test your program using several different inputs and check that it will always work correctly.

Document the program so that it can be used without studying the program code.

Programming style

Divide the problem into small, easily manageable and sufficiently simple subtasks that can be implemented as procedures.

Each procedure should perform one well defined operation.

Procedures should be as independent of each others as possible. Global variables should be used sparingly and only when it is really necessary. If a procedure has a lot of parameters, something might be wrong.

Avoid side effects of procedures. This includes also output; the procedure may be called very many times. Output may be optional for debugging. Also, do not scatter output all over the program.

Each program file should begin with a comment explaining the purpose and function of the file.

If modules in other files have to be linked to the program, the initial comments should contain the necessary compilation and linking commands.

In the initial comments, explain also what input files the program may need.

At the beginning of each procedure there should be a comment on the essentials of the procedure: what it will do, its parameters, posssible global variables, side effects and, if it a function, what value it will return.

The program must not ask for data that is not really needed but can be determined automatically. Format of the input should be as free and simple as possible. The output should be self-evident, understandable without reading the manual.

Test that the input data makes sense. The program must inform the user if the input is incorrect and recover from problems if possible. Be prepared for unexpected end-of-file situations.

To some variables, reasonable default values may be given.

The output should contain also the input data (or at least some identification of it) so that it easy to see which nput is related to the output.

Testing

Use different input data to test your programs. Also remember limiting cases (like minimal and maximal data sets) and check that the program will work also in extreme cases.

Include the outputs of several test cases. There should be at least a few cases for which the result can be checked manually.

Investigate how the accuracy behaves in different situations. The potential user wants to know what precision can be expected.

Also investigate how the execution time depends on the input. This may require rather large input data.

Documentation

In this work it is most important to give proper instructions to the user. Think what you would need if you had to use a program that somebody else has made. The document must be detailed enough so that the user can use the program without studying the program code,

If the program is a subroutine library, explain what each procedure will do and how they are invoked.

Numerical methods used should be mentioned briefly but detailed description is not necessary if the methods are generally known. In case the methods are not very well-known, a more specific explanation should be given.

If the program will read files, explain their contents and structure.

All programs contain some practical constraints (size of data, number of variables etc.) Explain these. If they can be easily changed by changing constants in the programs, that should be explained.