We have presented a handful of tools for organizing the control flow and data structures of computer programs. We have also shown what can happen if not enough care is taken in these tasks. This is not to say that these are the only control constructs that lead to intelligible programs. Nor can we guarantee that using just these structures will yield a readable program. But it helps.

The specific points of this chapter are:

(1) Write your program first in a made-up high-level language that you like, where you can see and debug your algorithm. When it is correct and well done, translate it into whatever language you have a compiler for.

(2) The control-flow constructions in your pseudo language should include:

The ability to group statements, as in PL/I's DO-END or BEGIN-END.

IF-ELSE, where the ELSE part is optional.

CASE, which is a multi-way decision. In PL/I it can be written with a series of ELSE-IF's. In Fortran, the computed GOTO can sometimes serve.

DO-WHILE, which repeats a set of statements zero or more times while some condition is true. Note that the range of a Fortran DO is generally executed at least once, so the DO statement must be preceded by an IF and GOTO whenever a zero or negative repeat count might occur.

Subroutines and functions, to break your code into small, separate, manageable pieces.

Your translation should be based on these constructions. GOTO's and labels are suspect; use them sparingly. Any GOTO's and labels in the final product should reflect these constructions only.

(3) Plan your data structures with the same care that you use for the control flow. Try to find a data representation that leads to a simpler program.