Use `IF ... ELSE IF ... ELSE IF ... ELSE ...` to implement multi-way branches.

Use the fundamental control flow constructs.

Write first in an easy-to-understand pseudo-language; then translate into whatever language you have to use.

Avoid `THEN-IF` and null `ELSE`.

Avoid `ELSE GOTO` and `ELSE RETURN`.

Follow each decision as closely as possible with its associated action.

Use data arrays to avoid repetitive control sequences.

Choose a data representation that makes the program simple.

Don't stop with your first draft.

Modularize. Use subroutines.

Make the coupling between modules visible.

Each module should do one thing well.

Make sure every module hides something.

Let the data structure the program.

Don't patch bad code — rewrite it.

Write and test a big program in small pieces.

Use recursive procedures for recursively-defined data structures.

Test input for validity and plausibility.

Make sure input cannot violate the limits of the program.

Terminate input by end-of-file or marker, not by count.

Identify bad input; recover if possible.

Treat end of file conditions in a uniform manner.

Make input easy to prepare and output self-explanatory.

Use uniform input formats.

Make input easy to proofread.

Use free-form input when possible.

Use self-identifying input. Allow defaults. Echo both on output.

Localize input and output in subroutines.

Make sure all variables are initialized before use.

Don't stop at one bug.

Use debugging compilers.

Initialize constants with `DATA` statements or `INITIAL` attributes; initialize variables with executable code.

Watch out for off-by-one errors.

Take care to branch the right way on equality.