Principles

MacLennan's Principles

- **Abstraction**
  - Avoid requiring something to be stated more than once; factor out the recurring pattern.
  - Subprograms, user defined types, inheritance
- **Automation**
  - Automate mechanical, tedious, or error-prone activities.
  - Garbage collection; looping structures

MacLennan's Principles (2)

- **Defense in Depth**
  - Have a series of defenses so that if an error isn't caught by one, it will probably be caught by another.
  - Array bound being part of type; definite loops.
- **Information Hiding**
  - The language should permit modules to be designed so that (1) the user has all of the information needed to use the module correctly, and nothing more; and (2) the implementor has all of the information needed to implement the module correctly, and nothing more.
  - Modules, packages, objects

MacLennan's Principles (3)

- **Labeling**
  - Avoid arbitrary sequences more than a few items long. Do not require the user to know the absolute position of an item in a list. Instead, associate a meaningful label with each item and allow the items to occur in any order.
  - Case statement, position-independent parameters.
- **Localized Cost**
  - Users should only pay for what they use; avoid distributed costs.
  - Violations: Algol60 loops, dynamic type binding.

MacLennan's Principles (4)

- **Manifest Interface**
  - All interfaces should be apparent (manifest) in the syntax.
  - Module specifications; function prototypes
- **Orthogonality**
  - Independent functions should be controlled by independent mechanisms.
  - Algol68 types; Ada parameter passing

MacLennan's Principles (5)

- **Portability**
  - Avoid features or facilities that are dependent on a particular machine or a small class of machines.
  - Ada prohibition of aliasing; C/Algol60 I/O
- **Preservation of Information**
  - The language should allow the representation of information that the user might know and that the compiler might need.
  - Definite looping structures