Use static analysis tools, including Fortran compilers, to detect problematic code, such as

- Language features that are obsolescent, non-conforming, or deleted
- Uninitialized variables
- Integer overflows

Enable the compiler's detection of such code. 6.22, 6.25, 6.53, 6.56, 6.57, 6.54, 6.58

Enable bounds checking and pointer checking throughout development of a code and only disable such checking during production runs when performance requirements cannot be met otherwise. 6.8, 6.14

Use all run-time checks that are available during development to detect:

- Uninitialized variables
- Real value exceptions
- Integer overflows
- Null pointer checks
- Dangling pointer checks
- Recursion depth ???
  6.2, 6.52

Declare all variables and use `implicit none` to enforce this. 6.17, 6.54

Use allocatable arrays where array operations involving differently-sized arrays might occur so the left-hand side array is reallocated as needed. 6.8

Use allocatable objects in preference to pointer objects whenever the facilities of allocatable objects are sufficient. 6.14, 6.33, 6.39

Specify explicit interfaces

- by placing procedures in modules where the procedure is to be used in more than one scope,
- by using internal procedures where the procedure is to be used in one scope only, and
- for all external procedures invoked by ??? (John to supply)

6.11, 6.32, 6.53, 6.57

Do not use keywords as names and do not reuse names in nested scopes. 6.17, 6.20

Declare a function as pure whenever possible; otherwise replace it with a subroutine. 6.24

Cover cases that are expected never to occur with a case default clause to ensure that unexpected cases are detected and processed, perhaps emitting an error message. 6.27

Perform IO on any given file in one programming language only; consider restricting all IO to one language system only. 6.47

Specify argument intents to allow further checking of argument usage. 6.32

Avoid the use of the intrinsic function `transfer`. 6.53

Include an `iostat` variable in each IO statement and check its value to ensure no errors occurred. *6.6*

Avoid sequence types. 6.11

Use coarrays only when communication among images is necessary. 6.61

Avoid the use of the `volatile` attribute. 6.61

Avoid the use of the `sync memory` statement for defining and ordering segments. 6.61

Use collective subroutines whenever possible. 6.61, 6.63

Always use intent specifications for dummy arguments. 6.65

Use procedures from a trusted library to perform calculations where floating-point accuracy is needed. Understand the use of the library procedures and test the diagnostic status values returned to ensure the calculation proceeds as expected. 6.4

Comment EP: delete a few; there are too many yet.