

# Fortran 202Y Suggestions

July 22, 2022

WG5 members were asked to submit a list of five suggested features/changes for Fortran 202Y. This paper collates the suggestions received to date, with a first attempt to break out by subgroup. Where available, links to elaboration are provided. If more than one person suggested an idea, the number of submissions is noted and sorted to top. Other than that, no order is implied.

The following work items were deferred by WG5 from Fortran 2023 (202X) to the 202Y revision and are to be included:

- Generics/Templates (US-24 <https://j3-fortran.org/forum/viewtopic.php?f=17&t=16>)
- Protected components and types (US-19 <https://j3-fortran.org/forum/viewtopic.php?f=17&t=26>)
- Pointer intent (US-27 <https://j3-fortran.org/forum/viewtopic.php?f=17&t=30>)

## Data

1. (3) Delete default implicit typing [https://github.com/j3-fortran/fortran\\_proposals/issues/90](https://github.com/j3-fortran/fortran_proposals/issues/90)
2. (2) Delete implied SAVE for initialized variables [https://github.com/j3-fortran/fortran\\_proposals/issues/40](https://github.com/j3-fortran/fortran_proposals/issues/40)
3. Namespace for modules: [https://github.com/j3-fortran/fortran\\_proposals/issues/1](https://github.com/j3-fortran/fortran_proposals/issues/1)
4. (2) Make it easier to process assumed-rank arguments in Fortran code [https://github.com/j3-fortran/fortran\\_proposals/issues/144](https://github.com/j3-fortran/fortran_proposals/issues/144)
5. Obsolete (not delete) default implicit typing
6. [bf16](#) AND [fp16](#)
7. Somehow fix the issue of mixed precision
8. Augmented assignment (+=, etc.) [https://github.com/j3-fortran/fortran\\_proposals/issues/113](https://github.com/j3-fortran/fortran_proposals/issues/113)
9. Enable Polymorphic Outputs from Pure/Simple Procedures: [https://github.com/j3-fortran/fortran\\_proposals/issues/196](https://github.com/j3-fortran/fortran_proposals/issues/196)
10. [Revisit strings](#) fortran remains frustratingly bad at strings, surely we can do better for 202Y?!
11. [Default values for optional args](#)
12. Simple Functions in Constant Expressions: [https://github.com/j3-fortran/fortran\\_proposals/issues/253](https://github.com/j3-fortran/fortran_proposals/issues/253) and [Add CONSTEXPR procedures in Fortran · Issue #214 · j3-fortran/fortran\\_proposals \(github.com\)](#)
13. Option to derive an inextensible derived type <https://j3-fortran.org/doc/year/19/19-186.txt>
14. “Proper” enumeration types <https://j3-fortran.org/doc/year/19/19-229.txt>
15. Review all the restrictions on polymorphic entities and remove as many as is reasonable.
16. [Program-specified default KINDs for constants and intrinsics · Issue #78 · j3-fortran/fortran\\_proposals \(github.com\)](#)

## JOR

1. (6) Standardize cpp-like preprocessor [https://github.com/j3-fortran/fortran\\_proposals/issues/65](https://github.com/j3-fortran/fortran_proposals/issues/65)

2. (2) Better error handling [https://github.com/j3-fortran/fortran\\_proposals/issues/6](https://github.com/j3-fortran/fortran_proposals/issues/6)
3. (2) scan / prefix sum [https://github.com/j3-fortran/fortran\\_proposals/issues/273](https://github.com/j3-fortran/fortran_proposals/issues/273)
4. Obsolete D format edit descriptor [https://github.com/j3-fortran/fortran\\_proposals/issues/226](https://github.com/j3-fortran/fortran_proposals/issues/226)
5. Disallow use of specific (standard provides a list) new-to-2Y features in a program unit that also uses a deprecated or deleted feature [https://github.com/j3-fortran/fortran\\_proposals/issues/280](https://github.com/j3-fortran/fortran_proposals/issues/280)
6. Surprising results of LBOUND and UBOUND when argument has zero extent [https://github.com/j3-fortran/fortran\\_proposals/issues/254](https://github.com/j3-fortran/fortran_proposals/issues/254)
7. Change floating point model to reflect IEEE 754 so that intrinsic examples aren't as surprising. [https://github.com/j3-fortran/fortran\\_proposals/issues/268](https://github.com/j3-fortran/fortran_proposals/issues/268)
8. log2 [https://github.com/j3-fortran/fortran\\_proposals/issues/222](https://github.com/j3-fortran/fortran_proposals/issues/222)
9. Go thru all the processor dependencies, and try to eliminate as many as practical, without undue burden on the implementors or host/target operating system, ... Perhaps add something similar to ISO\_ENV stuff to give the user compile/runtime knowledge of what are now processor dependencies in some cases.
10. [Intrinsics to return details of where call originated](#) doable with macros, but horrid!
11. [ASSERT](#) would be nice.
12. A shorthand for immutability: [https://github.com/j3-fortran/fortran\\_proposals/issues/221](https://github.com/j3-fortran/fortran_proposals/issues/221)
13. "scan clause for do concurrent reduce" [https://github.com/j3-fortran/fortran\\_proposals/issues/224](https://github.com/j3-fortran/fortran_proposals/issues/224)
14. BIT data type (BITS proposal from <https://j3-fortran.org/doc/year/19/19-159.txt> )

## HPC

1. (3) additional collective subroutines and optional TEAM variable [https://github.com/j3-fortran/fortran\\_proposals/issues/223](https://github.com/j3-fortran/fortran_proposals/issues/223)
2. fetch-and-op atomics in DO CONCURRENT fetch-and-op atomics in DO CONCURRENT [https://github.com/j3-fortran/fortran\\_proposals/issues/270](https://github.com/j3-fortran/fortran_proposals/issues/270)
3. asynchronous blocks / tasks (equivalent to OpenACC async and OpenMP non-dependent tasks)
4. (2) <https://github.com/llvm/llvm-project/blob/main/flang/docs/DoConcurrent.md> documents problems with DO CONCURRENT. These problems have also been discussed at [https://github.com/j3-fortran/fortran\\_proposals/issues/62](https://github.com/j3-fortran/fortran_proposals/issues/62) and in J3 paper 19-134 (<https://j3-fortran.org/doc/year/19/19-134.txt> ).
5. allow immediate return from collective subroutines [https://github.com/j3-fortran/fortran\\_proposals/issues/272](https://github.com/j3-fortran/fortran_proposals/issues/272)
6. co\_scan {inclusive, exclusive} [https://github.com/j3-fortran/fortran\\_proposals/issues/223](https://github.com/j3-fortran/fortran_proposals/issues/223)
7. Refer to polymorphic entities on other images: [https://github.com/j3-fortran/fortran\\_proposals/issues/251](https://github.com/j3-fortran/fortran_proposals/issues/251)
8. Asynchronous/task-based parallel features: [https://github.com/j3-fortran/fortran\\_proposals/issues/274](https://github.com/j3-fortran/fortran_proposals/issues/274)
9. Extend collective semantics <https://j3-fortran.org/doc/year/22/22-163.txt>

## Edit

1. Make an effort to re-organize the standard to make it more readable. I don't know if this is practical, and maybe we don't have enough editorial resources to do this, but a small piece at a time approach is likely the best way to start and see if this suggestion is viable.
2. Add a second project editor. This is generally a good practice, even if the primary editor continues to do most of the work.