Fortran 202Y Suggestions

July 19, 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.

Note: Several people suggested generics/templates, but this is already accepted for 2Y by WG5

Data

1. (3) Delete default implicit typing 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
3. Namespace for modules: 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
5. Obsolete (not delete) default implicit typing
6. Pointer intent (deferred from 202X) https://github.com/j3-fortran/fortran_proposals/issues/5
7. bf16 AND fp16
8. Somehow fix the issue of mixed precision
9. Augmented assignment (+=, etc.) https://github.com/j3-fortran/fortran_proposals/issues/113
10. Enable Polymorphic Outputs from Pure/Simple Procedures: https://github.com/j3-fortran/fortran_proposals/issues/196
11. Revisit strings fortran remains frustratingly bad at strings, surely we can do better for 202Y?!
12. Default values for optional args
13. Simple Functions in Constant Expressions: https://github.com/j3-fortran/fortran_proposals/issues/253
14. Option to derive an inextensible derived type https://j3-fortran.org/doc/year/19/19-186.txt
15. "Proper" enumeration types https://j3-fortran.org/doc/year/19/19-229.txt
16. Review all the restrictions on polymorphic entities and remove as many as is reasonable.

JOR

1. (5) Standardize cpp-like preprocessor https://github.com/j3-fortran/fortran_proposals/issues/65
2. (2) Better error handling https://github.com/j3-fortran/fortran_proposals/issues/6
3. (2) scan / prefix sum https://github.com/j3-fortran/fortran_proposals/issues/273
4. Obsolete D format edit descriptor 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
6. Surprising results of LBOUND and UBOUND when argument has zero extent 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
8. log2 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
13. "scan clause for do concurrent reduce" https://github.com/j3-fortran/fortran_proposals/issues/224
14. Program-specified default KINDs for constants and intrinsics · Issue #78 · j3-fortran/fortran_proposals (github.com)
15. Add CONSTEXPR procedures in Fortran · Issue #214 · j3-fortran/fortran_proposals (github.com)

HPC

1. fetch-and-op atomics in DO CONCURRENT fetch-and-op atomics in DO CONCURRENT https://github.com/j3-fortran/fortran_proposals/issues/270
2. asynchronous blocks / tasks (equivalent to OpenACC async and OpenMP non-dependent tasks)
3. 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 and in J3 paper 19-134 (https://j3-fortran.org/doc/year/19/19-134.txt).
4. allow immediate return from collective subroutines https://github.com/j3-fortran/fortran_proposals/issues/272
5. additional collective subroutines https://github.com/j3-fortran/fortran_proposals/issues/223
6. co_scan {inclusive, exclusive} https://github.com/j3-fortran/fortran_proposals/issues/223
7. co_reduce_scatter
8. co_gather
9. co_scatter
10. Refer to polymorphic entities on other images: https://github.com/j3-fortran/fortran_proposals/issues/251
11. Asynchronous/task-based parallel features: https://github.com/j3-fortran/fortran_proposals/issues/274
12. 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.