

Questions for “The future of HPC programming - a Modern Fortran workshop”, 2022-11-(24-25)

You are welcome to ask questions in the Zoom chat as well, but it easily gets cluttered when many are using it at the same time. You can instead write your question here, and the helpers will then either answer it or read it to the presenter so he can answer.

Please note the following:

- Add your questions below, numbering them continuously.
- Be careful in case someone else is writing at the same time.
- Please **DO NOT** delete your questions even when they have been answered, as we are planning to use them to improve our material.
- Do not share any sensitive information as this document is accessible to anyone with the correct link.
- You can start a new line by pressing <SHIFT> + <ENTER>.

Day 1 / Thursday

- 1) General information / summary:
https://umeauniversity.sharepoint.com/:w:/s/HPC2N630/Ec8-loVmUCNKuhj_-QuLYqUBW_cuOz_CY_QP_wya3fBMgA?e=krjKrC
- 2) For those that are using HPC2N's resources
 - a) Hostnames for kebnekaise login nodes:
 - i) **SSH:** `kebnekaise.hpc2n.umu.se`
 - ii) **ThinLinc:** `kebnekaise-t1.hpc2n.umu.se`
 - b) How to change the password?
 - i) After logging in, type `passwd`. It will first ask your **current** password and then your **new** password twice. There might be a small delay before the new password has been communicated across the system.
 - c) In order to access the Fortran compilers, you need to load a module. We recommend the module `foss/2021b` (contains GCC 11.2) or `foss/2022a` (contains GCC 11.3). Either module can be loaded directly (do a “`module purge`” in between if you are switching between them).
 - i) To load `foss/2021b`, type: `m1 foss/2021b`
 - ii) To load `foss/2022a`, type: `m1 foss/2022a`
 - iii) When the chosen module is loaded, you can access the compilers, like `gfortran`.
- 3) [BalintAradi]: Will the slides eventually be available for download?
 - a) We haven't discussed that yet – we will get back.
 - b) **Update:** Jonas will make slides and codes available. Expecty email from Birgitte.

- 4) [BA] Is it not a “dangerous” practice to make the `init()` method type bound? As in Fortran everything is virtual, you won’t be able to override its signature in any extending types.
- Did this answer your question?
 - Partially yes, thanks. But the problem is, that the signature remains the same. If an extending type would need to pass further arguments `init()`, it will not be possible.
 - But isn’t this against the grain of object orientation?
 - Depends on your framework. Think about Circle and Rectangle. Circle needs only a radius to get initialized, while Rectangle two lengths, `a` & `b`.
 - But then you would not build circle on rectangle or vice versa. You would take a point (e.g. centre of gravity) and build the circle or the rectangle on top of the point with the extra attribute.
 - That’s true, but then you need two calls to initialize an object, which is error prone. What happens,
 - Jonas suggested to call the initialiser to the point from the initialiser of e.g. the square. So if I want to initialise the square I call the initialiser of the square, that’s it.
 - Yes, you’re absolutely right. The problem is only, that you can not pass any other data to the `init()` of square, as to the `init()` of point. But square may need also additional data to initialize itself, so you would have pass it with an extra additional call.:
 - My understanding is, that square `init` and point `init` are different routines and can have different arguments.
 - Unfortunately, that does not work in Fortran. Extending type must override the procedures with the same signature (apart of the first `class(...)` argument. /
 - I try to catch Jonas on this later ...
 - I think he is demonstrating this now ...
 - Exactly: You can not change the signature of a call in extending types. This is why we usually make the initialization not a class-method in our Fortran projects.
/
- 5) [MD] Why do you use “`type(...) :: this`” and sometimes “`Class(...) :: this`” inside the subroutines.

When should I use `Class` and when `type`?

- [BA] In all type bound procedures, you use `class()`, unless in the finalizer, where you must use `type()`. /
- 6) Practical question: will there be any breaks before lunch?
- There will be a break around 10:00 or 10:15. The break will be about 15 min
- 7) An example is GPU computing.
- 8) I think, there is no casting. If `c_int` can not be represented in Fortran (if there is no corresponding type in Fortran), the compiler would stop with an error message. /
- 9) If you want to see what happens when types are not compatible try passing Logical from Fortran to boolean in C/C++.
- What would you use in Fortran to match on a C/C++ boolean?
 - Don’t, pass integers instead

- 10) Are you on a windows system? Could you share, after the class, a note on how to get the environment for mixed (Fortran and C/C++) programming up and running?
- 11) Very, very nice for the GUI capabilities, thanks a lot for demonstrating, but if I didn't miss it, you had no multidimensional arrays in the Fortran code, which arguably would be a main reason to code the "number crunching" part in Fortran. How to pass multidimensional arrays from C++ to Fortran? In general a Fortran library would require multidimensional arrays as inputs.
- a) Fortran multidimensional arrays are passed as flat arrays to C/C++ and then back. A (N, M) Fortran array becomes a C/C++ array with N*M elements.
 - > ok, thanks, but that scales bad for higher dimensions, like rank 4, 5, 6... I mean from the coding point of view, you need to take care manually of all the strides?
 - i) In C/C++ yes, that sucks.
 - > So eigen, boost, etc. do not provide "official" ISO bindings?
 - ii) C++23 will include mdspan, which should make this easier.
 - b) One has to watch out for column major or row major formatting of multi dimensional arrays in Fortran/C
 - c)

12) , Do you lose noticeable performance running this simulation in C++ and Fortran over just writing all in C++? Or is it even faster?

13) It is possible to combine MPI-codes. You just pass the MPI-communicator handle around, converting it between the Fortran and the C++ representation using the appropriate MPI-routines. /

 - a) I think this is a question towards the MPI standard.
 - b) It is explicitly supported by the MPI standard.

14) We have break until 13:00!

15) [BA] Amazing and very nice! Are there any plans to make the notation less verbose, before being standardized? Currently it feels to be a lot of typing.

16) Did you already say which Fortran compiler is giving all this facility (Generic Programming)?

 - a) <https://fortran.org/> is doing proto-typing

17) [BA]: Also the NAG compiler takes 1, 2, 3 by default as kinds for the real kinds.

 - a) Tack.

18) I am confused here. If we have to instantiate ourselves, what is the point? Isn't the idea of templates that the compiler instantiates the right types for us? ^^


 - a) We will get to that ;)

19) Perhaps, start with something simple? Just addition 😊? We can continue..

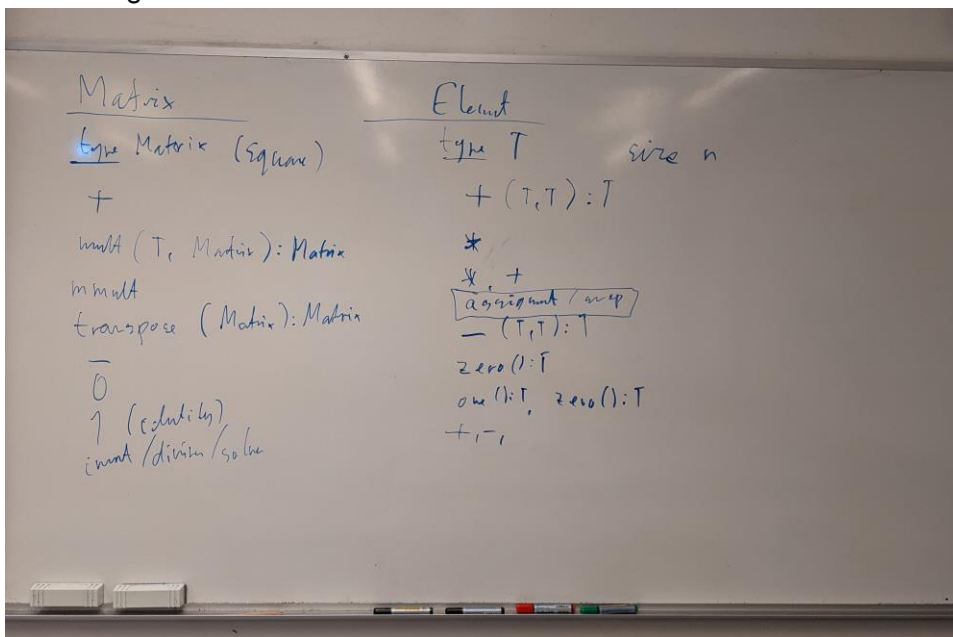
20) Exercise:

Exercise: templated Matrix library

- Which types are involved in a matrix library?
 - Defined types
 - Required types
- Which operations should a matrix library have?
 - Defined types
- What are the requirements for each such operations?
 - Required operations

Magne Haveranen • Future of HPC programming: Fortran evolution • <https://bidl.i.uib.no>

Answering:



21) I believe this is the link to the YouTube video about GraphBLAS:

<https://youtu.be/wqjRzC2fPUo>


22) [BA] Actually, the current NAG compiler supports co-arrays (with thread parallel execution).

23) We reconvene at 16:10

24) Exercise:

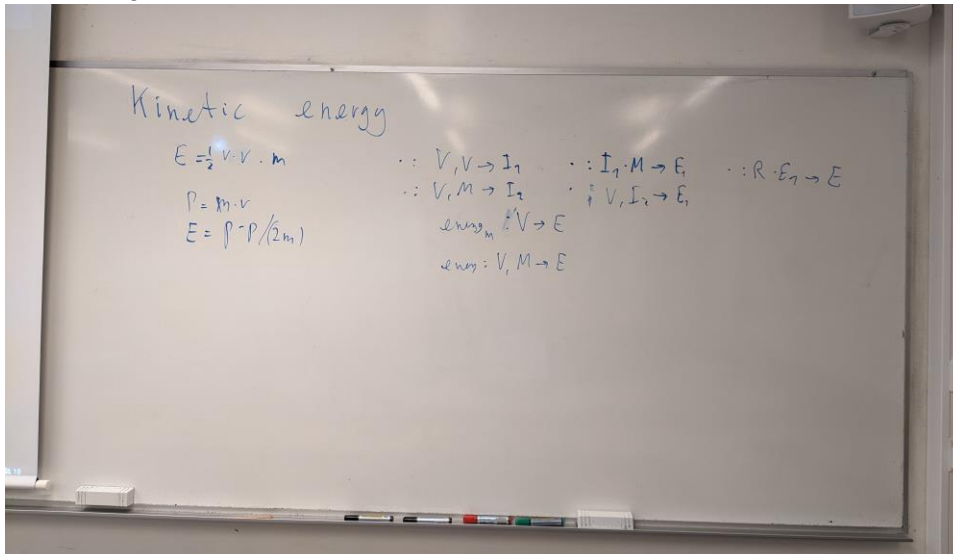
Exercise: Problem specific requirements

- Define the requirements for
 - Kinetic energy
 - Speed
 - Mass
 - Energy
 - Momentum
 - Potential energy
 - Converting between kinetic and potential energy
- Define templates doing these computations



Magne Haveråen • Future of HPC programming: Fortran evolution • <https://bldl.it.uib.no>

Answering/some ideas:



25) [BA] A follow-up technical question: Would one be able to distribute the templates in a library as binary file, or does one have to distribute the template source code? Thx, very cool!

26) Start tomorrow at 8:30 CET

27) PLACEHOLDER

28) PLACEHOLDER

29) PLACEHOLDER



30) PLACEHOLDER

Day 2 / Friday

1. [BA] I think, you could also invent automatic reindexing arrays during the transformation, to map information, which is not present for all entries... (as Magne explained....)
2. So is the compiler like modern gfortran versions doing the transformation AoS to SoA and the code is almost same speed? I did not understand if it is a real problem.
 - a. It is a real problem. Current compilers do not do the transformation.
 - b. Ok thanks for the answer. I am not very experienced, does anyone know if this is an issue in c++ or python as well, or is it fortran specific, since for example the c++ compiler does this optimization?
 - c. [JH]I it a general problem with derived data types in OO. I am not aware of any major compiler/runtime engine helping you.
 - d. [BA] Compiler use the data layout you tell them. They are not reoptimizing your approach. That would be one level up above current compilers. /
 - e. Also, different operations call for different optimal memory layouts. So what should the compiler do, reshape memory every time? It doesn't sound so optimal neither... I believe it's highly nontrivial of a task.
3. The sound quality is dramatically better when Magne is nearer to the laptop :-)
4. Exercise time until 10:40 ...

Exercise: Develop an IDSL for your domain

- Find the types used in that domain
 - Often not manifest in how we speak about the domain
 - Manifest in the way we think about the domain
- Find the operations used in that domain
 - Often not manifest in how we speak about the domain
 - Manifest in the way we think about the domain
- Make certain everything is made explicit
 - The intention of all types
 - The precise intention of all operations
 - All input information needed for each operation
 - All output information from the operation
 - We can easily be confused by words/names

Magne Haverajen • Future of HPC programming: Fortran evolution • <https://bldl.il.uib.no>

5. Will we get the slides?
 - a. Magne will be sharing some slides later, most likely via Birgitte
6. How about CUDA and modern Fortran in the future?

7. [BA] Non-initilaized data is bug indeed. However, the question is: do you prefer a buggy program with indeterministic behavior (when no default intiialization is done) or a buggy program with deterministic behavior (if default initialization is done). Reproducible results would for sure need latter.
8. [LV] For automatic storage duration, can't you initialize with f.ex. `int a[200] = {0}`, even in C? Data ought to be consistent between the stack memory and cache lines, I've never seen it depend on whether it was hot or cold.
9. [LV] (more of a comment) On the discussion on the C++ list, there's a suggestion to zero-initialize memory on the stack for automatic variables and it is shown to often even improve performance (<https://wg21.link/P2723R0>)!
10. [LV] You have an OS guarantee on pretty much every OS to get memory backed by zero pages when allocating fresh memory. Allocators inside the process like malloc gets you old values if reusing previously freed but not returned memory.
11. [BA] Does it mean, that libraries would be distributed as ASRs, you would combine ASRs during build and then create machine code?
12. I came back late from lunch – when will these recordings be posted?
 - a. Depends on Birgitte's laptop doing the mangling
 - b. Either later today or during the weekend. Should definitely be up before Monday
13. For anyone to join who might want to take part of the discussion on the Fortran language, on development, future, etc.:
 - a. <https://fortran-lang.discourse.group/>
14. [BA] Thank you for organizing this very interesting workshop!
15. PLACEHOLDER
16. PLACEHOLDER
17. PLACEHOLDER
18. PLACEHOLDER
19. PLACEHOLDER
20. PLACEHOLDER
21. PLACEHOLDER
22. PLACEHOLDER
23. PLACEHOLDER
24. PLACEHOLDER
25. PLACEHOLDER