codee

Automated Inspection of Fortran/C/C++ Code Using Codee for Correctness, Modernization, Optimization, and Security on HPE/Cray

CUG 25 Tutorial

May 5, 2025

Manuel Arenaz: manuel.arenaz@codee.com

Agenda

1. Codee for Simulation Software
2. Codee Formatter & Demos
3. Codee Analyzer & Demos
4. Want to Know more?
5. Hands-on

Agenda

1. **Codee for Simulation Software**
2. Codee Formatter & Demos
3. Codee Analyzer & Demos
4. Want to Know more?
5. Hands-on

Your Main Drivers for Simulation Software?

1 Ensure **correctness** , **modernization** , **portability** and **security**

Mandatory

- **Modern programming best practices** make code easier to understand and to work with, helping **prevent bugs**.
- **Use cases:**
  - **Catch bugs**.
  - Enforce project-specific **coding guidelines.**
  - **Modernize legacy code**; e.g.: **F77 → F2018, C++98 → C++20**.
  - **Ensure portability** across compilers; no vendor-specific language extensions.
  - **Address security vulnerabilities**.
  - **Migration to new programming environments; e.g. port from Intel ifort to ifx.**
  - **Replace in-house ad-hoc Fortran analyzers** difficult to maintain and develop.
  - Automated testing in **CI/CD frameworks**; e.g.: GitLab, GitHub Actions, Jenkins.
  - Other use cases; e.g.: **32-bit → 64-bit code.**

2 Consider addressing **optimization** only when needed

Optional

- **Address performance optimization at the end of the coding process**, once all of the correctness, modernization & security issues have been fixed.

Main Features

From **F77** up to **F2023**

Fixed-form and **free-form**

Standard Fortran Language and Compiler **Non-Standard Dialects** (GNU, Intel)

Code Formatting

Enforce a uniform source code format to write cleaner, more consistent, and maintainable code effortlessly.

Static Analysis

Automatically analyze every line of code to find and fix modernization and optimization opportunities.

Reports

Get a deeper understanding of your code's health with analysis reports.

Autofix

Automatically generate fixes for opportunities, always under the control of the programmer and preserving 100% code correctness.

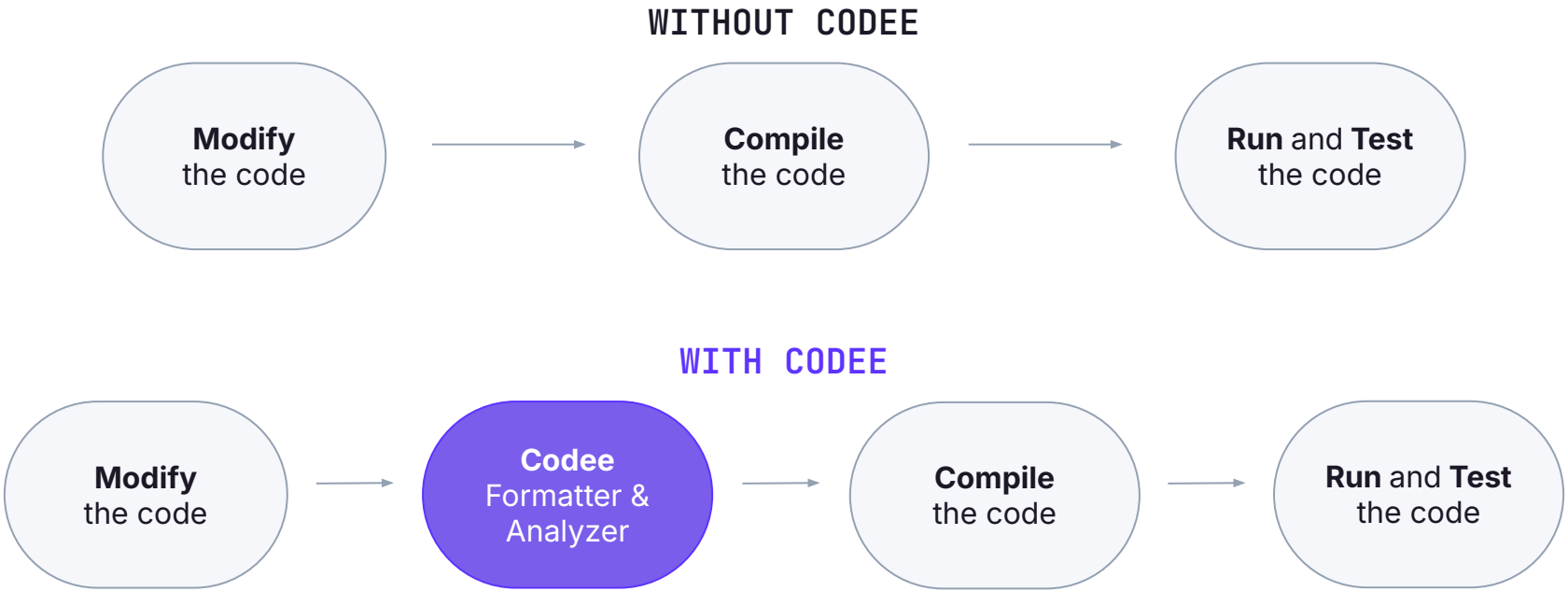
CI/CD automation

Integrate with CI/CD systems, automatically testing every code change and pull request.

Self-hosting

Execution on the local system, retraining full control of your code and privacy.

Developer Workflow powered by codee



Codee provides a systematic, predictable workflow that is a complement to other software development tools, such as the compiler, profiler, or debugging tools.

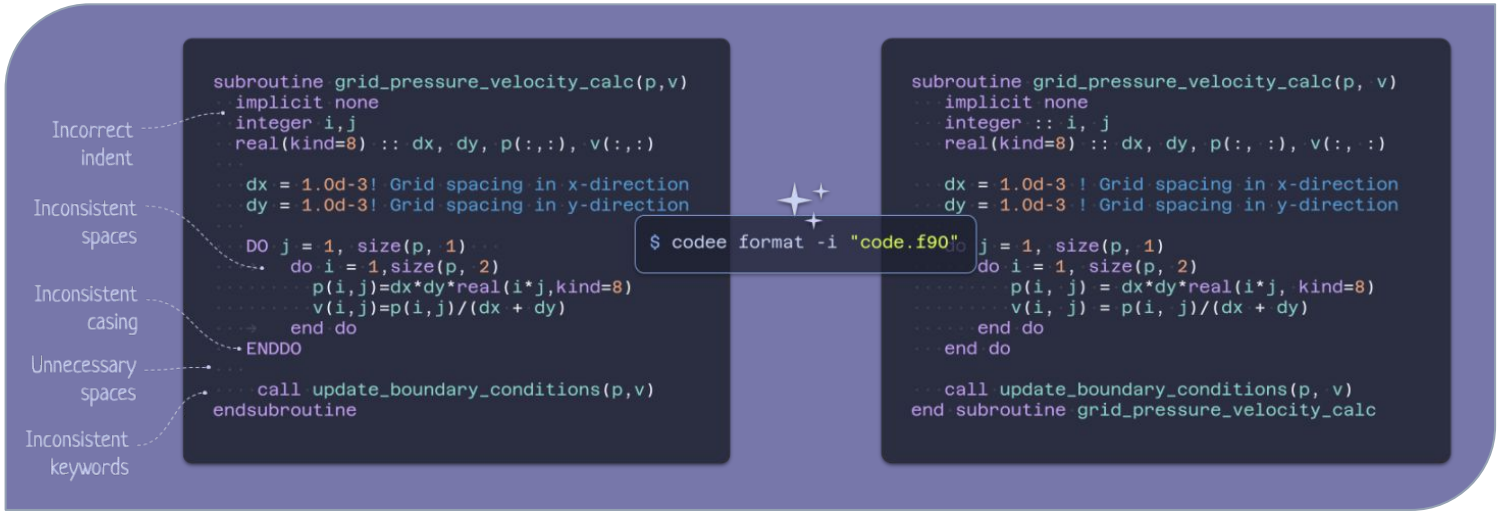
Agenda

1. Codee for Simulation Software
2. **Codee Formatter & Demos**
3. Codee Analyzer & Demos
4. Want to Know more?
5. Hands-on

Formatter for Fortran

https://www.codee.com/codee-formatter

- **Highly-customizable free-form formatting:**
  - Enforce a consistent style across the codebase
  - +20 style options → docs.codee.com/formatter/style-options
- **[Upcoming feature] Fixed-form formatting**



CODEE

Formatter for Fortran

Feature name	Feature description	Codee	fprettify	fortitude	findent
Command-line interface (CLI) Integration	Command-line simple usage, integrable into editors or CI pipelines	✔	✔	✔	✔
Config File Customization	Customization file for code style enforcement with extensive documentation	✔	✘	✔	✘
Partial File Formatting	Format only parts of the code, ideal for IDE selections or git commits	✔	✘	✘	✘
Format Suppression Comments	Suppression of formatting in code sections with comments	✔	✔	✔	✔
Detailed Documentation	Up-to-date detailed documentation with all the options explained	✔	✘	✔	✔
Integration Guides	Step-by-step guides with integration with git, the most used IDEs and CI pipelines	✔	✘	✘	!
Modern Fortran Support	Support up to the latest version of Fortran (2023)	✔	✘	✔	✔
Automatic Line Indentation	Automatic indentation based on code scope	✔	✔	✘	✔
Column Limit	Breaking long lines into multiple smaller ones	✔	✘	!	✘
Operator Spacing Consistency	Consistent spacing around operators and keywords	✔	✔	✘	✘
Uniform Operator Style	Choose between symbolic or literal representation of Fortran operators	✔	✔	✔	✘
Consistent Keyword Casing	Ensure consistent casing of keywords, identifiers and operators	✔	✔	✘	✘
End Statement Style	Ensure joined/separated, with/without names end statements	✔	✘	✔	✔
Double-Colon in Declarations	Add missing double-colon token to variable declaration	✔	✘	✔	✘
Kind Keyword Enforcement	Enforcing the usage of the kind keyword in intrinsic declaration	✔	✘	✘	✘
Split Multiline Statements	Break each statement into a separate line	✔	✘	✘	✘
Remove Superfluous Semicolons	Removes unnecessary semicolons	✔	✘	!	✘
Whitespace Cleanup	Redundant EOL, trailing whitespace and consecutive whitespaces	✔	✔	✔	✔
EOL Normalization	Enforce consistent end-of-line (LF or CRLF)	✔	✘	✘	✘

Command-Line Interface (CLI)

```
Usage:
  codee format [OPTIONS] <file>... <directory>...

Arguments:
  <file>
          One or more Fortran source files to be formmated. Source files must
          be free-form (e.g., `.f90`, `.f95`, `.f03`, `.f08`)

  <directory>
          One or more directories containing Fortran source files. All Fortran
          files within the specified directories will be recursively formatted

Command options:
      --config <file>
          Set the path to the configuration file. It defaults to a
          `.codee-format` file in the directory of the input file or the current
          working directory if no input file is provided)

      --dump-config
          Dump the effective configuration options to stdout and exit. It considers
          configuration files and command-line options

  -o, --out-file <file>
          Apply the modifications in a new output file

      --style, --style=<string>
          Set coding style. `string` is a JSON-like string `"{key: value, ...}"`
          that overrides specific parameters of the configuration file
```

Simple command-line interface for terminal, editors and CI frameworks

Applicable to files and directories

Config File Customization

Codee Formatter Live demo

1. Command-line usage
2. Git integration
3. VSCode integration
4. CI/CD integration

Agenda

1. Codee for Simulation Software
2. Codee Formatter & Demos
3. **Codee Analyzer & Demos**
4. Want to Know more?
5. Hands-on

Analyzer for Fortran/C/C++

- **Identify improvement opportunities across the codebase:**
  - Enforce correctness, modernization, security, portability, and optimization
  - +80 checks → open-catalog.codee.com
- **AutoFix opportunities:**
  - Automatically apply suggestions
  - Preserve 100% correctness

```
$ codee checks --verbose --check-id PWR072

thornado/SandBox/TwoMoment_OrderV/Executables/../InitializationModule.F90:397:27 [PWR072] (level: L1):
Split the variable initialization from the declaration to prevent the implicit 'save' behavior
  Suggestion: Split the initialization of the variable 't_0' from its declaration to prevent storing the
              value across different invocations of the procedure; in case this behavior is intended, add
              the 'save' attribute to the variable to make the intent explicit
  Documentation: https://github.com/codee-com/open-catalog/tree/main/Checks/PWR072
  AutoFix:
    codee rewrite --check-id pwr072 --in-place \
    thornado/SandBox/TwoMoment_OrderV/Executables/../InitializationModule.F90:InitializeFields_SphericalDiffusion

<...>
```

Feedback from the community

"Always use IMPLICIT NONE everywhere. ***It is amazing how many bugs this can find and avoid*** *compared to the default typing rules."*

...

"All subprograms should be CONTAINed. Generally in modules, but also in the main program unit. <...> ***Again, amazing how many interface bugs show up when this is enforced****."*

...

Open Catalog of Code Guidelines

Fortran/C/C++ checks used by the Codee Analyzer:
open-catalog.codee.com

Checks

All checks | Correctness | Modernization | Security | Portability | Optimization

Filter checks...

ID	Title	Category	C	Fortran	C++	AutoFix
PWR001	Declare global variables as function parameters	correctness, modernization, security	✓	✓	✓	
PWR002	Declare scalar variables in the smallest possible scope	correctness, security	✓		✓	
PWR003	Explicitly declare pure functions	modernization, security, optimization	✓	✓	✓	
PWR004	Declare OpenMP scoping for all variables	correctness	✓	✓	✓	
PWR005	Disable default OpenMP scoping	correctness	✓	✓	✓	
PWR006	Avoid privatization of read-only variables	optimization	✓	✓	✓	
PWR007	Disable implicit declaration of variables	correctness, modernization, security		✓		✓[1]
PWR008	Declare the intent for each procedure parameter	correctness, modernization, security		✓		✓[1]
PWR009	Use OpenMP teams to offload work to GPU	optimization	✓	✓	✓	
PWR010	Avoid column-major array access in C/C++	optimization	✓		✓	
PWR012	Pass only required fields from derived type as parameters	modernization, optimization	✓	✓	✓	

Command-Line Interface (CLI)

```
Usage:
  codee <command> [OPTIONS] <filter>... --compile-commands <compile_commands>
  codee <command> [OPTIONS] <filter>... -- <compiler invocation>

Commands:
  checks
          Report opportunities, recommendations and other actionable items found
          in the input(s)

  format
          Format Fortran code

  rewrite
          Apply an AutoFix

  screening
          Print a screening report of the given input(s)

Arguments:
  <compiler invocation>
          The command used to invoke the compiler. Examples:
              clang -I path/to/include test.c
              gfortran foo.f90 bar.f90

Common options:
  -p, --compile-commands <filepath>
          Load the analysis targets options from the specified path. The path
          can either be a JSON compilation database file (`compile_commands.json`)
          or a directory that contains a file named `compile_commands.json`
```

Codee “linter” for Fortran/C/C++

Codee Formatter

Codee’s AutoFixes to modify the source code

Prioritization report with ranking of checkers

Simple command-line interface with JSON compilation database

Options to Select Target Files/Directories

```
Usage:
  codee <command> [OPTIONS] <filter>... --compile-commands <compile_commands>
  codee <command> [OPTIONS] <filter>... -- <compiler invocation>

Arguments:
  <filter>
          Determine which parts of the inputs will be analyzed. It is composed
          of a filepath, followed by an optional list of function names or
          specific positions in the file. For specifying positions, use the
          format `line number:column number`. Use commas to separate items in
          the list. For instance:
              path/to/file.ext:foo,bar
              test.c:3:2,2

Common options:

      --exclude <file|directory>
          Skip the specified file or directory. `--exclude` may be set several
          times

     --lang <language>
          Filter the input files by language (C, C++, Fortran)
```

Specify the target source code, including files, directories, procedures and line numbers

Exclude selected files/directories

Options to Select Subsets of Checkers

```
Common options:

      --check-id <id>[,<id>]*
          Only enable the checks that match the specified ID(s). When used with
          the `rewrite` subcommand, it applies the autofixes specified by the
          given check ID(s)

      --no-check-id <id>[,<id>]*
          Exclude the checks that match the specified ID(s). When used with the
          `rewrite` subcommand, it disables the autofixes specified by the given
          check ID(s). Takes precedence over all other check filters

      --level <L1|1|high|L2|2|medium|L3|3|low>
          Filter the checks by priority level

      --list-available-checkers
          List all available defects, recommendations and remarks

      --include-categories <category>[,<category>]*
          Enable the checks that match the specified categories, in addition to
          those enabled by default

      --only-categories <category>[,<category>]*
          Enable the checks that match the specified categories only

      --target-arch <arch>
          Filter the checks by target architecture
```

Enable selected checkers

Disable selected checkers

Enable checkers by priority level

List details about the checkers

Codee Checks

```
$ codee checks himeno.f90
. . .
CHECKS REPORT

himeno.f90 [PWR063] (level: L1): Avoid using legacy Fortran constructs
himeno.f90:136:1 [PWR001] (level: L3): Declare global variables as function parameters
himeno.f90:164:1 [PWR001] (level: L3): Declare global variables as function parameters
himeno.f90:223:1 [PWR001] (level: L3): Declare global variables as function parameters
himeno.f90:255:1 [PWR001] (level: L3): Declare global variables as function parameters
himeno.f90:275:1 [PWR001] (level: L3): Declare global variables as function parameters

1 file, 7 functions, 5 loops successfully analyzed and 0 non-analyzed files in 188 ms



$ codee checks --verbose --check-id pwr063 himeno.f90
. . .
himeno.f90 [PWR063] (level: L1): Avoid using legacy Fortran constructs
  PAUSE:
    131:   pause
  Suggestion: Remove the legacy fortran constructs and refactor the code to comply with modern Fortran standards.
  Documentation: https://github.com/codee-com/open-catalog/tree/main/Checks/PWR063
. . .
1 file, 7 functions, 5 loops successfully analyzed and 0 non-analyzed files in 145 ms
```

Codee Screening with Ranking

```
$ codee screening himeno.f90
. . .
SCREENING REPORT

Lines of code Analysis time # checks Profiling
------------- ------------- -------- ---------
214           194 ms        10       n/a

CHECKS PER CATEGORY AND PRIORITY LEVELS

| ---------------Checks per category---------------- | Priority |
| Scalar Control Memory Vector Multi Offload Quality | L1 L2 L3 |
| ------ ------- ------ ------ ----- ------- ------- | -- -- -- |
| 0      0       2      2      n/a   n/a     6       | 3  0  7  |

RANKING OF CHECKERS

Checker Level Priority # Title
------- ----- -------- - ----------------------------------------------------------
RMK015  L1    P27      1 Tune compiler optimization flags to increase the speed of the code
PWR054  L1    P12      1 Consider applying vectorization to scalar reduction loop
PWR063  L1    P12      1 Avoid using legacy Fortran constructs
PWR001  L3    P3       5 Declare global variables as function parameters
PWR035  L3    P2       2 Avoid non-consecutive array access to improve performance

1 file, 7 functions, 5 loops successfully analyzed and 0 non-analyzed files in 195 ms
```

Total number of checkers triggered

Checkers per category/priority

List of checkers reported, ordered by priority

Codee Rewrite (I)

First, run Codee to produce the Checks Report in verbose mode. For those checks that have AutoFix capabilities, the tool will suggest invocations of the *codee rewrite* command.

```
$ codee checks --verbose --check-id pwr051 himeno.f90
. . .
himeno.f90:293:6 [PWR051] (level: L2): Consider applying multithreading parallelism to scalar reduction loop
  Suggestion: Use 'rewrite' to automatically optimize the code
  Documentation: https://github.com/codee-com/open-catalog/tree/main/Checks/PWR051
  AutoFix (choose one option):
      * Using OpenMP 'for' with built-in reduction (recommended):
        codee rewrite --multi omp-for --in-place himeno.f90:293:6
      * Using OpenMP 'for' with explicit privatization:
        codee rewrite --multi omp-for --in-place --explicit-privatization gosa himeno.f90:293:6
      * Using OpenMP 'taskwait':
        codee rewrite --multi omp-taskwait --in-place himeno.f90:293:6
      * Using OpenMP 'taskloop':
        codee rewrite --multi omp-taskloop --in-place himeno.f90:293:6

. . .

1 file, 7 functions, 5 loops successfully analyzed and 0 non-analyzed files in 145 ms
```

Codee Rewrite (II)

Second, run Codee to annotate the source code with OpenMP multithreading directives. The tool will provide details about the actual changes implemented in the source code.

```
$ codee rewrite --multi omp-for --in-place himeno.f90:293:6
. . .
Results for file 'himeno.f90':
  Successfully applied AutoFix to the loop at 'himeno.f90:jacobi:293:6' [using multi-threading]:
      [INFO] himeno.f90:293:6 Parallel scalar reduction pattern identified for variable 'gosa' with associative,
commutative operator '+'
      [INFO] himeno.f90:293:6 Parallel forall: variable 'wrk2'
      [INFO] himeno.f90:293:6 Available parallelization strategies for variable 'gosa'
      [INFO] himeno.f90:293:6   #1 OpenMP scalar reduction (* implemented)
      [INFO] himeno.f90:293:6   #2 OpenMP atomic access
      [INFO] himeno.f90:293:6   #3 OpenMP explicit privatization
      [INFO] himeno.f90:293:6 Loop parallelized with multithreading using OpenMP directive 'for'
      [INFO] himeno.f90:293:6 Parallel region defined by OpenMP directive 'parallel'

Successfully updated himeno.f90

Minimum software stack requirements: OpenMP version 3.0 with multithreading capabilities
```

Codee Rewrite (and III)

Finally, review the source code comparing the original code and the optimized code. The tool just adds annotations of OpenMP directives. As a coding assistant tool does not replace proper testing and benchmarking of the optimized code on your target hardware.

```
! Codee: Loop modified by Codee (2024-04-29 11:40:52)
! Codee: Technique applied: multithreading with 'omp-for' pragmas
!$omp parallel default(none) shared(a, b, bnd, c, gosa, imax, jmax, kmax, p, wrk1, wrk2) private(i, j, k, s0, ss)
!$omp do private(i, j, s0, ss) reduction(+: gosa) schedule(auto)
do k=2,kmax-1
   do j=2,jmax-1
      do i=2,imax-1
         s0=a(I,J,K,1)*p(I+1,J,K) &
              +a(I,J,K,2)*p(I,J+1,K) &
              +a(I,J,K,3)*p(I,J,K+1) &
              +b(I,J,K,1)*(p(I+1,J+1,K)-p(I+1,J-1,K) &
                          -p(I-1,J+1,K)+p(I-1,J-1,K)) &
              +b(I,J,K,2)*(p(I,J+1,K+1)-p(I,J-1,K+1) &
                          -p(I,J+1,K-1)+p(I,J-1,K-1)) &
              +b(I,J,K,3)*(p(I+1,J,K+1)-p(I-1,J,K+1) &
                          -p(I+1,J,K-1)+p(I-1,J,K-1)) &
              +c(I,J,K,1)*p(I-1,J,K) &
              +c(I,J,K,2)*p(I,J-1,K) &
              +c(I,J,K,3)*p(I,J,K-1)+wrk1(I,J,K)
         ss=(s0*a(I,J,K,4)-p(I,J,K))*bnd(I,J,K)
         GOSA=GOSA+SS*SS
        wrk2(I,J,K)=p(I,J,K)+OMEGA *SS
      enddo
    enddo
enddo
!$omp end parallel
```

Codee Analyzer Live demo

1. PWR079 detection
2. PWR072 detection + rewrite

Agenda

1. Codee for Simulation Software
2. Codee Formatter & Demos
3. Codee Analyzer & Demos
4. **Want to Know more?**
5. Hands-on

Want to Know More About Codee ?

Codee online documentation

Quickstarts:	https://docs.codee.com/quickstarts
Codee Formatter:	https://docs.codee.com/formatter
Codee Analyzer:	https://docs.codee.com/analyzer
Changelog:	https://docs.codee.com/changelog
FAQs:	https://docs.codee.com/faqs
Webinars:	https://www.codee.com/webinars/

Codee Online Documentation for the Platform HPE Cray EX (AMD EPYC CPU)

Introduction:	https://docs.codee.com/platforms/hpe-cray-ex
Step-by-step guides:	https://docs.codee.com/platforms/hpe-cray-ex/benchmarking
Automated testing:	https://docs.codee.com/platforms/hpe-cray-ex/cray-auto-testing/

Codee to Find Correctness Bugs

https://codee.com/wp-content/uploads/Correctness-in-Fortran-using-Codee.pdf

Codee to Enforce Coding Guidelines

https://codee.com/wp-content/uploads/Coding-Guidelines-in-Fortran-using-Codee.pdf

Codee for Fortran Security to Find Vulnerabilities

https://codee.com/wp-content/uploads/Security-in-Fortran-using-Codee.pdf

Agenda

1. Codee for Simulation Software
2. Codee Formatter & Demos
3. Codee Analyzer & Demos
4. Want to Know more?
5. **Hands-on**

Hands-on

1. **Get familiar with Codee using the step-by-step instruction of the Quickstart guides:**
   - Codee installation: https://docs.codee.com/quickstarts/installation
   - Codee Analyzer: Basic Workflow: https://docs.codee.com/quickstarts/Fortran-basic
   - Codee Analyzer: Added-value reporta: https://docs.codee.com/quickstarts/added-value-reports
   - Platform HPE Cray EX: https://docs.codee.com/platforms/hpe-cray-ex
2. **Follow the step-by-step instructions of the demos conducted with the Thornado source code:**
   - Step-by-step Thornado: *<see PDF distributed with the tutorial materials>*
3. **Start using Codee on your own source code!**

Contact me while at CUG 2025 for further discussion!

Manuel Arenaz

manuel.arenaz@codee.com

codee

Thank you!

Contact:
support@codee.com

www.codee.com

info@codee.com

Subscribe: codee.com/newsletter/

Spain

codee_com

/codee-com/