codee

Exploring the Challenges of the World-Class HPE Cray Programming Environment for Modern Software Development in Fortran

Programming Environments, Applications, and Documentation (PEAD)

May 5, 2025

Manuel Arenaz: manuel.arenaz@codee.com

Outline

- **Background**
- **Modern Software Development in Fortran**
- **Cray Programming Environment (CPE)**
- **Codee**
  - Use Cases in Modern Software Development in Fortran
  - Examples of Open Source Fortran/C/C++ Codes
  - Codee Formatter & Codee Analyzer
  - Integration with Compilers of the CPE
  - Highlights of the latest Codee 2025.2 (Apr 2025)
- **Codee & Fortran Security**
- **Want to Know More about Codee?**
- **Contact us at CUG 2025 !**

Background

- Latest release is Codee **2025.2.1** (Apr 22nd, 2025)
  - Release policy: Quarterly (Public, Announcement), Monthly (Internal, Delivery to customers/partners)
  - Changelog available at https://docs.codee.com/changelog
- ORNL has been **funding the development of Codee** for 4+ years, since 2020.
  - Main contact: David Bernholdt <bernholdtde@ornl.gov>
- ORNL request for 2024-2025: **Integration of Codee with HPE Cray tools**.
  - Develop Codee support for Fortran leveraging the LLVM Flang front-end
  - Develop Codee integrations for the Cray compilers
- Codee successfully **installed in HPE Cray systems** over the years:
  - HPE Cray systems at ORNL, NERSC, KAUST, PAWSEY,...
  - Many other systems at DKRZ, BSC,... including Linux, Windows and MacOS
- Currently working on an **HPE-Codee Partnership** at different levels:
  - Technical: Development of Codee integrations for Cray CPE and Cray CCE
  - Awareness: Activities in Cray Users Group (CUG) and in customer's training programs
  - Commercial: Offering of Codee as a part of the HPE Cray EX product lines
  - Legal: World-wide HPE-Codee Agreement under review

Modern Software Development in Fortran

Modern software development in Fortran has evolved significantly in the recent years, having key requirements that go beyond performance and also cover portability, maintainability, and security.

1. **Adherence to Modern Fortran Standards** (modernize legacy Fortran using Fortran 90/95/2003/2008/2018/2023).
2. **Modern Compiler Suites** (take advantage of the latest features and optimizations of modern Fortran compilers).
3. **Modular, Maintainable and Clean Code** (encapsulation, OO programming, up-to-date documentation, consistent coding style).
4. **Leverage Optimized Libraries** (LAPACK, BLAS, FFTW, HDF5).
5. **Optimize the Code for Better Performance** (OpenMP, MPI, OpenACC, CUDA, OpenCL, SYCL,...).
6. **Interfacing with Other Languages** (C, Python).
7. **Testing and Debugging** (unit testing frameworks, debuggers, static and dynamic analysis tools).
8. **Cross-Platform Development** (portability, Linux/Windows/macOS, built tools -CMake-, cross-compilers).
9. **Deployment and Packaging** (job schedulers -slurm-, containers -docker-, build tools -CMake, Makefiles, FPM-).
10. **Code Quality and Refactoring** (code review, code refactoring, static analysis).
11. **Bugs must be detected as soon as possible in the development process** (static analysis).
12. **Version Control and Collaboration** (Git, GItLab, GitHub,...).
13. **Continuous Integration** (CI pipelines).
14. **Security considerations**.
15. **Adoption of DevOps and DevSecOps best practices**.

Cray Programming Environment (CPE)

The HPE Cray Programming Environment (CPE) is a comprehensive suite of development tools designed to harness the full power of Cray supercomputers. It a powerful ecosystem for developing and optimizing modern scientific and engineering software at scale:

- **Cray Compiling Environment (CCE)**: Optimized compilers for Fortran, C/C++
- **Performance Analysis Tools**: Utilities like CrayPat for profiling and tuning application performance
- **Scientific Libraries**: Pre-optimized math and scientific computation libraries tailored for Cray architectures
- **Parallel Programming Support**: Built-in tools for MPI and OpenMP to enable shared and distributed memory models
- **DevOps Integration**: Support for CI/CD pipelines, containerization, and workflow automation

It is key to streamline the development process while ensuring that Fortran software can scale, perform efficiently, and remain secure.

Developers will benefit from an "enhanced" Cray Programming Environment (CPE) that offers new tools for Fortran:
- Static Analyzer for Fortran
- Code Formatter for Fortran
- Static Application Security Testing (SAST) for Fortran
- Software Composition Analysis (SCA) for Fortran

codee

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.

Autofix

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

Reports

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

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.

Use Cases in Modern Software Development in Fortran

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.

Examples of Open Source Fortran/C/C++ Codes

Code	Domain	Metrics with Codee 2025.2.1 (Apr 2025)
CP2K 1.3M lines of code	Quantum chemistry and solid state physics software package	`1361 files (291 with missing deps), 11306 functions, 22399 loops, 1093520 LOCs successfully analyzed (20829 checkers) and 0 non-analyzed files in 1 h 12 m 47 s`
OpenRadioss 1.1M lines of code	Finite element solver for dynamic event analysis	`3519 files, 6692 functions, 39742 loops, 1132227 LOCs successfully analyzed (39412 checkers) and 0 non-analyzed files in 1 h 15 m 0 s`
WRF 960K lines of code	Weather Research and Forecasting	`508 files, 9700 functions, 26246 loops, 949428 LOCs successfully analyzed (75118 checkers) and 0 non-analyzed files in 5 h 17 m 34 s`
ICON 646K lines of code	Weather, climate, and environmental prediction	`1154 files, 11694 functions, 21644 loops, 655567 LOCs successfully analyzed (14788 checkers) and 0 non-analyzed files in 23 m 41 s`
SIESTA 398K lines of code	First-principles Materials Simulation	`1683 files (699 with missing deps), 8330 functions, 8100 loops, 656212 LOCs successfully analyzed (13814 checkers) and 0 non-analyzed files in 6 m 42 s`
PHASTA 64K lines of code	Parallel Hierarchic Adaptive Stabilized Transient Analysis of compressible and incompressible Navier Stokes equations	`284 files (6 with missing deps), 705 functions, 1408 loops, 63051 LOCs successfully analyzed (2595 checkers) and 0 non-analyzed files in 19 m 35 s`
HYCOM 44K lines of code	Hybrid Coordinate Ocean Model	`50 files, 250 functions, 2107 loops, 45242 LOCs successfully analyzed (1740 checkers) and 0 non-analyzed files in 1 m 11 s`
EAP-patterns 4K lines of code	Patterns from an Eulerian cell AMR application	`12 files, 88 functions, 164 loops, 3724 LOCs successfully analyzed (141 checkers) and 0 non-analyzed files in 3128 ms`

CODEE

Formatter

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)	✔	✘	✘	✘

CODEE

Analyzer

Codee is a key enabler to **identify and quantify the technical debt** required to enhance correctness, modernization, security, portability, and optimization of the **Fortran/C/C++ files**.

OpenBLAS 1M lines of code 2K files in Fortran 3K files in C	OpenBLAS is an optimized BLAS library based on GotoBLAS2 1.13 BSD version.	`5193 files, 7502 functions, 18583 loops, 1035170 LOCs successfully analyzed (32200 checkers) and 70 non-analyzed files in 27 m 25 s`

Checker	Category	Priority	AutoFixes	#	Title
PWR079	correctness, portability, security	P27 (L1)		13	Avoid undefined behavior due to uninitialized variables
PWR063	correctness, modern, security	P12 (L1)		715	Avoid using legacy Fortran constructs
PWR068	correctness, modern, security	P9 (L2)		13224	Encapsulate procedures within modules to avoid the risks of calling implicit interfaces
PWR008	correctness, modern, security	P9 (L2)	9	2338	Declare the intent for each procedure argument
PWR007	correctness, modern, security	P9 (L2)	2190	2193	Disable the implicit declaration of variables and procedures
PWR069	correctness, modern, security	P9 (L2)	16	16	Use the keyword only to explicitly state what to import from a module
PWR003	modern, security, other	P6 (L2)		78	Explicitly declare pure functions
PWR018	security, control	P6 (L2)		6	Call to recursive function within a loop inhibits vectorization
PWR071	modern, portability, security	P3 (L3)		7135	Prefer real(kind=kind_value) for declaring consistent floating types
PWR002	correctness, security	P3 (L3)		3376	Declare scalar variables in the smallest possible scope
PWR037	correctness, security	P3 (L3)		13	Potential precision loss in call to mathematical function
PWR073	correctness, modern, security	P3 (L3)		3	Transform common block into a module for better data encapsulation
PWR070	correctness, modern, security, memory	P2 (L3)		2143	Declare array dummy arguments as assumed-shape arrays
PWR028	security, control	P2 (L3)		717	Remove pointer increment preventing performance optimization
PWR030	security, control	P2 (L3)		2	Remove pointer assignment preventing performance optimization for perfectly nested loops
PWR001	correctness, modern, security	P1 (L3)		228	Pass global variables as function arguments
Total			2215	32200	

Online documentation for the automated testing of OPENBLAS:
https://docs.codee.com/platforms/azure/azure-gh-runner

Integration with Compilers of the CPE

Codee feature	GNU	LLVM	Intel	Cray	Nvidia	AMD
Codee CLI built-in support for the compiler model (i.e. flags, include paths)	gcc g++ gfortran	clang clang++ clang-cl flang	icc icpc lfort icx icpx ifx	craycc craycxx crayCC crayftn ftnfe	nvfortran	clang flang
Codee's bundled bear tool extended with built-in support for the compiler model (i.e. invocation, wrappers, flags)	✔	✔	✔	✔	✔	✔
Codee CLI built-in support for widely-used Fortran modules (e.g. OpenMPI, HDF5)	✔	✔	✔	✔	✔	✔
Codee CLI built-in file dependency manager due to usage of Fortran modules	✔	✔	✔	✔	✔	✔
Codee CLI built-in support for non-standard Fortran extensions of the compiler, including intrinsic functions	✔		✔			
Codee CLI built-in support to analyse the compiler's optimization reports (advanced feature "codee diagnose --compiler-efficiency")	✔	✔	✔			
Codee's checkers identify portability issues due to non-standard Fortran extensions of the compiler	✔					
Codee's AutoFix generates OpenMP pragmas	✔	✔	✔	✔	✔	✔
Codee's AutoFix generates OpenMP offloading pragmas tuned for the compiler	✔			✔	✔	
Codee's AutoFix generates compiler-specific vectorization pragmas	✔	✔	✔			
Codee's AutoFix generates OpenACC pragmas	✔			✔	✔	
Codee's AutoFix generates Fortran native `do concurrent`	✔	✔	✔	✔	✔	✔

Codee 2025.2 (Apr 2025)

- **Target market is Simulation Software using Fortran/C/C++**
- **Codee Formatter (NEW)**
  - The most advanced tools for Formatting source code written in Fortran
  - Online documentation with step-by-step guides, integrations, etc (https://docs.codee.com/formatter)
- **Codee Analyzer**
  - Automated detection of 86 checkers for Correctness, Modernization, Portability, Security and Optimization
  - Automated Autofixes of 4 checkers of Correctness & Modernization (e.g. *implicit*, *intent*, *use only*, *save*)
  - Automated Autofixes of 9 checkers of Optimization (e.g. multithreading, offload, vectorization, OpenMP, OpenACC, GNU/Intel/LLVM pragmas, Fortran `do concurrent`)
  - Online documentation with step-by-step guides, integrations, etc (https://docs.codee.com/analyzer)
- **Open Catalog**
  - Online documentation browsable at https://open-catalog.codee.com/
  - Initiative intended for collaboration with the community through https://github.com/codee-com/open-catalog
  - PWR079 (undef), PWR072 (save)
  - PWR007 (implicit), PWR008 (intent), PWR069 (use only), PWR068 (implicit interfaces), PWR070 (assume-shaped arrays),
  - PWR075 (non-standard Fortran by GNU/Intel)
  - ...
- **Major improvements in User eXperience (UX)**
  - **Integration with compilers** GNU, LLVM, Intel (Classic, LLVM), HPE/Cray, NVIDIA, AMD, Microsoft
  - **Deep file dependencies analysis** related to Fortran modules, external libraries and #include statements
  - **Bundled extended Bear tool to generate compilation databases** (compile_comands.json)
  - **Bundled widely used Fortran modules** (e.g. ISO_FORTRAN_ENV, OpenMPI, NetCDF, HDF5, OpenACC, OpenMP, libmath, CBLAS)
  - **Distributed for Linux & Windows, compatible with docker containers**.
  - **Reports exported to JSON, CSV, SARIF and HTML**.

Codee & Fortran Security

- **Recent cybersecurity regulations affect simulation software written in Fortran/C/C++**
  - National Institute of Standards and Technology. (2022). Secure Software Development Framework (SSDF) Version 1.1 (NIST Special Publication 800-218). https://doi.org/10.6028/NIST.SP.800-218 [Accessed Apr 21, 2025].
  - European Commision. (2024). Cyber Resilience Act. https://digital-strategy.ec.europa.eu/en/policies/cyber-resilience-act [Accessed Apr 21, 2025].
- **Why is security important in Fortran now?**
  - Fortran was considered "safe by obscurity"; scientific, engineering, and HPC software not exposed to the internet or hostile environments.
  - Fortran software still powers critical infrastructures (e.g., weather prediction, nuclear simulations, aerospace, defense).
  - Cybersecurity attacks have shown that any piece of code can become an attack target if it is part of a larger system.
  - Compliance requirements from governments and large enterprises now make vulnerability scans and secure coding practices mandatory.
  - Fortran code is part of critical digital infrastructure, so even 40-year-old numerical code must comply with cybersecurity requirements.
- **New need for tools that enable consistent, efficient, and scalable application of security practices across complex codebases:**
  - Static Application Security Testing (SAST), which analyzes source code for vulnerabilities.
  - Software Composition Analysis (SCA), which identifies risks in third-party dependencies.
- **Codee tools provide the all-in-one solution for Fortran, leveraging its Deep Analysis for Fortran to help with the automation of security best practices.**

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

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

Contact us at CUG 2025!

Technical Program:
https://ssl.linklings.net/conferences/cug/cug2025_program/views/at_a_glance.html

Day	Technical program activity	
Sunday 4:00pm - 4:30pm (30') *Newport I*	**BoF Programming Environments, Applications, and Documentation (PEAD):** Presentation of the talk "Exploring the Challenges of the World-Class HPE Cray Programming Environment for Modern Software Development in Fortran".	
Monday 1:00pm - 2:30pm 2:30pm - 4:15pm (3h) *Newport I*	**Tutorial 2B:** Title "Automated Inspection of Fortran/C/C++ Code Using Codee for Correctness, Modernization, Optimization, and Security on HPE/Cray".	**Sponsor booth!**
Monday 1:00pm - 2:30pm (1h30') *Newport IV*	**BoF 1D, Security BoF:** New HPC Security Special Interest Group for exchanging and discussing strategies and ideas related to ensuring the secure configuration, operation and utilization of Cray/HPE systems, which includes Codee's capabilities for Fortran Security.	
Wednesday 10:35am - 10:45am (10') *Newport III*	**Plenary Sponsors Talks:** Title "Codee: A Tool to Enhance Correctness, Modernization, Security, Portability and Optimization in Fortran and C/C++ Software Applications".	
Thursday		

Manuel Arenaz
manuel.arenaz@codee.com

www.codee.com

info@codee.com

Subscribe: codee.com/newsletter/

Spain

codee_com

/codee-com/