GSoC, 2023
Program @CERN-HSF

Student : Smit Shah

Mentors : Vassil Vassilev, Baidyanath Kundu

Enable cross-talk between Python and C++ kernels in xeus-clang-REPL by using Cppyy

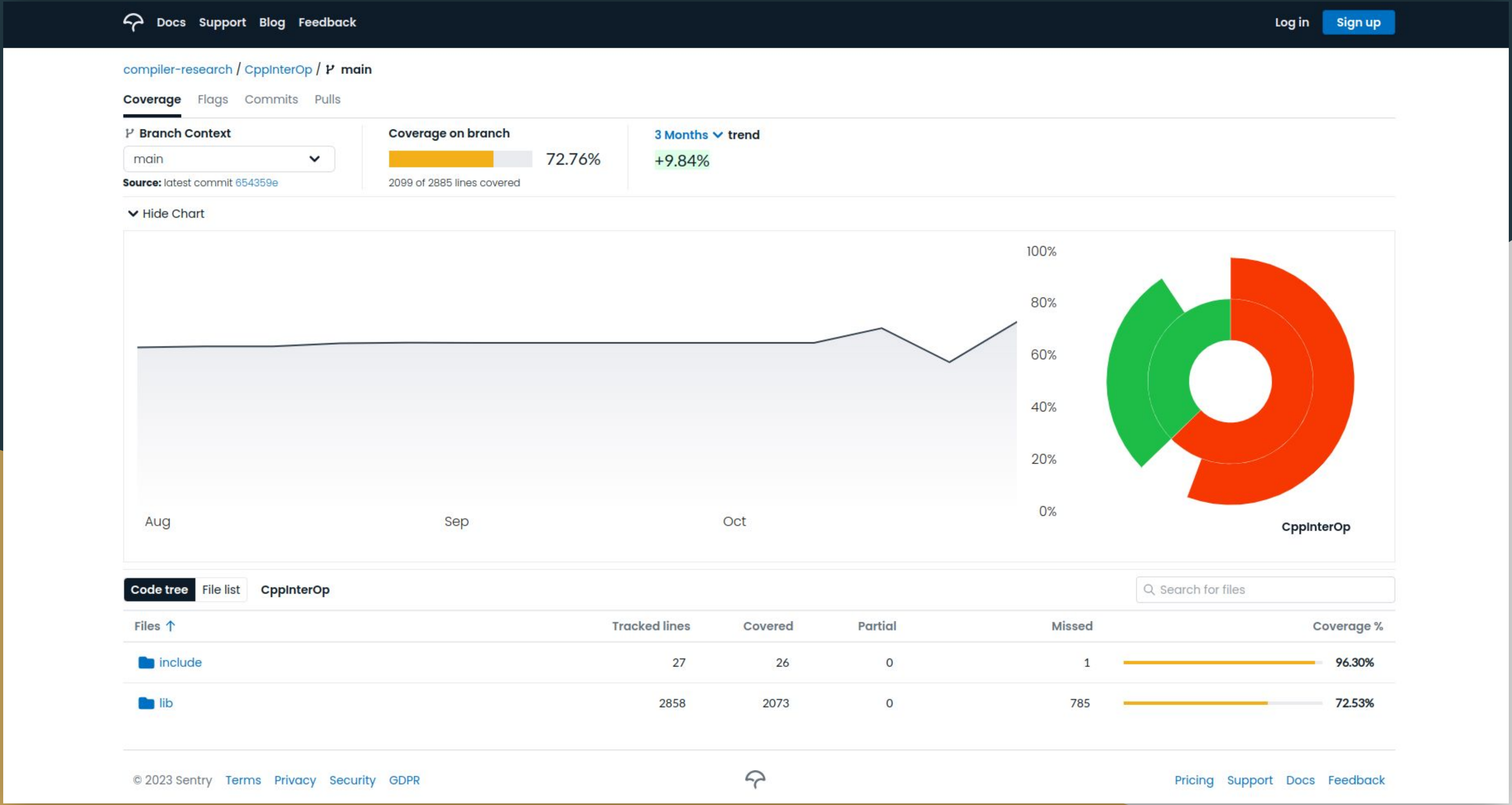
Summary of my work done

Initial Phase

1] In the initial phase of the coding period, emphasis was placed on implementing basic functionalities for the CppInterOp repository.

2] These functionalities included the generation of Code Coverage reports and the integration of clang-tidy and clang-format checks within the GitHub Continuous Integration (CI) pipeline.

3] These initiatives were undertaken to improve the overall code quality and maintain compliance with established coding standards.


Docs Support Blog Feedback Log in Sign up

compiler-research / CppInterOp / main

Coverage Flags Commits Pulls

Branch Context
main
Source: latest commit 654359e

Coverage on branch
72.76%
2099 of 2885 lines covered

3 Months trend
+9.84%

Hide Chart

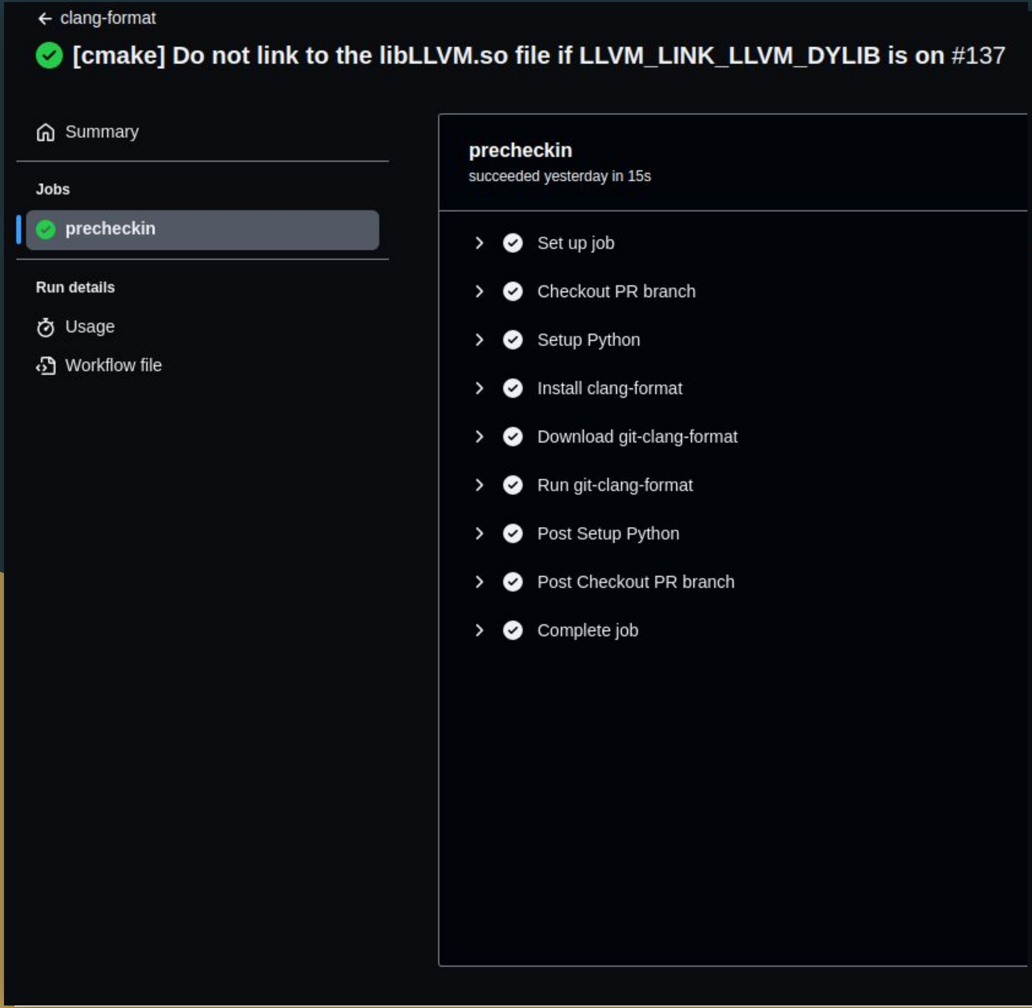
Code tree File list CppInterOp

Search for files

Files	Tracked lines	Covered	Partial	Missed	Coverage %
include	27	26	0	1	96.30%
lib	2858	2073	0	785	72.53%

© 2023 Sentry Terms Privacy Security GDPR

Pricing Support Docs Feedback


← clang-format

[cmake] Do not link to the libLLVM.so file if LLVM_LINK_LLVM_DYLIB is on #137

Summary

Jobs

precheckin

Run details

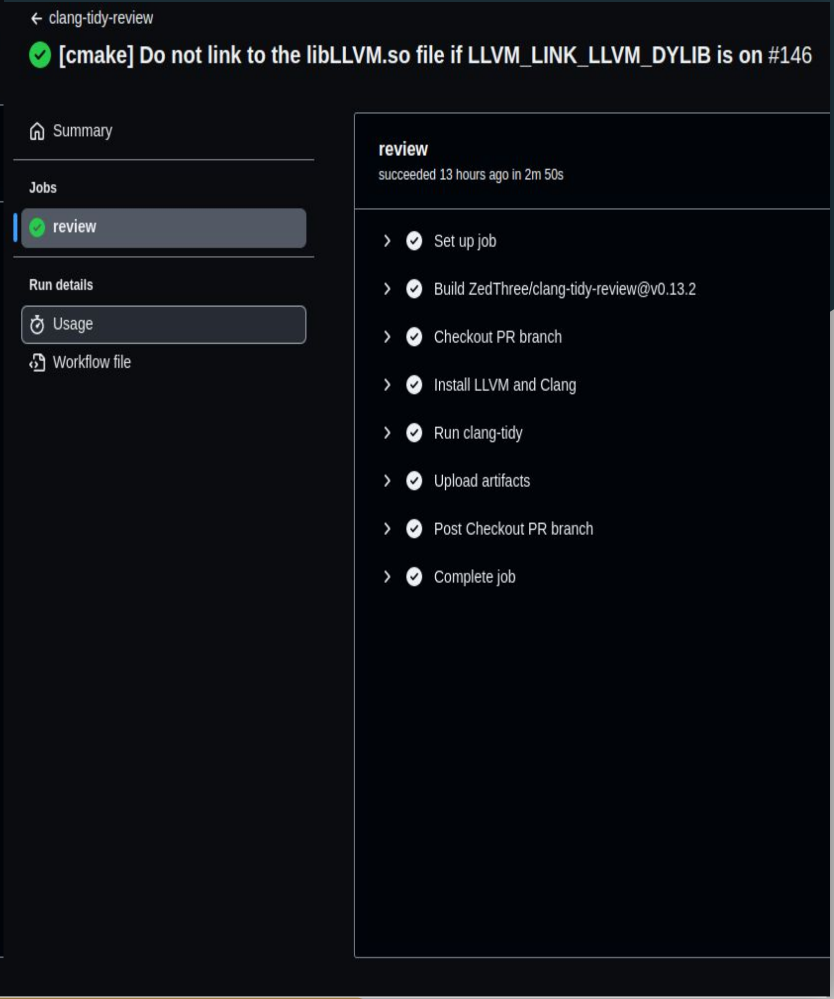
Usage

Workflow file

precheckin

succeeded yesterday in 15s

- Set up job
- Checkout PR branch
- Setup Python
- Install clang-format
- Download git-clang-format
- Run git-clang-format
- Post Setup Python
- Post Checkout PR branch
- Complete job


← clang-tidy-review

[cmake] Do not link to the libLLVM.so file if LLVM_LINK_LLVM_DYLIB is on #146

Summary

Jobs

review

Run details

Usage

Workflow file

review

succeeded 13 hours ago in 2m 50s

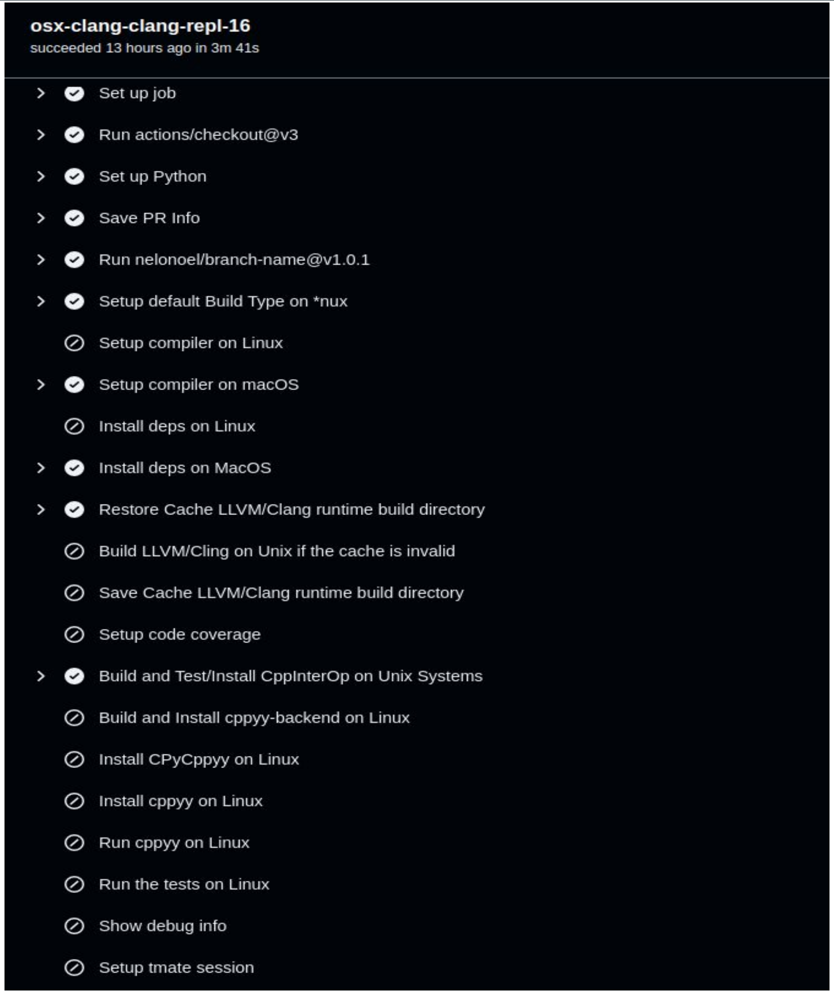
- Set up job
- Build ZedThree/clang-tidy-review@v0.13.2
- Checkout PR branch
- Install LLVM and Clang
- Run clang-tidy
- Upload artifacts
- Post Checkout PR branch
- Complete job

Mid Phase

1] Originally, CppInterOp was confined to Ubuntu-based platforms for its build and testing processes.

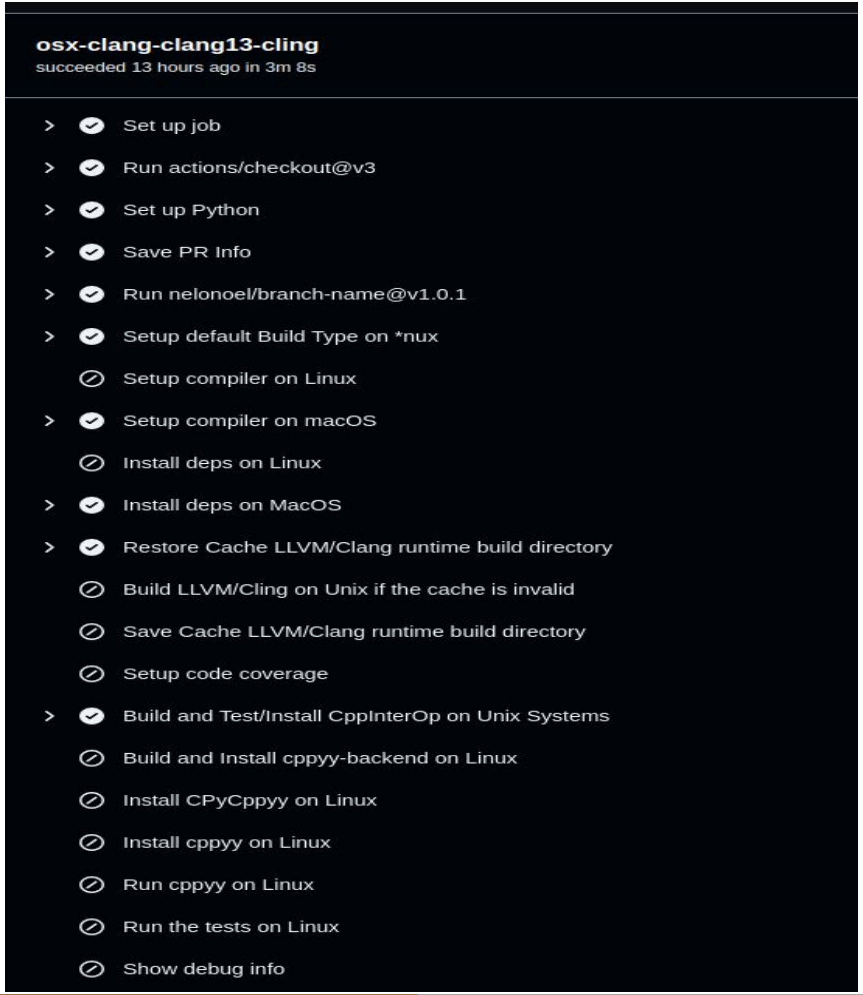
2] To expand its usability, a build structure was implemented for macOS. This structure was successfully integrated into the GitHub Continuous Integration (CI) system.

3] The current priority is on developing a similar build structure for Windows. This initiative aims to make CppInterOp easily deployable and accessible across various platforms, ensuring its widespread usability.



osx-clang-clang-repl-16
succeeded 13 hours ago in 3m 41s

- ✓ Set up job
- ✓ Run actions/checkout@v3
- ✓ Set up Python
- ✓ Save PR Info
- ✓ Run nelonoel/branch-name@v1.0.1
- ✓ Setup default Build Type on *nux
- ⊘ Setup compiler on Linux
- ✓ Setup compiler on macOS
- ⊘ Install deps on Linux
- ✓ Install deps on MacOS
- ✓ Restore Cache LLVM/Clang runtime build directory
- ⊘ Build LLVM/Cling on Unix if the cache is invalid
- ⊘ Save Cache LLVM/Clang runtime build directory
- ⊘ Setup code coverage
- ✓ Build and Test/Install CppInterOp on Unix Systems
- ⊘ Build and Install cppyy-backend on Linux
- ⊘ Install CPyCppyy on Linux
- ⊘ Install cppyy on Linux
- ⊘ Run cppyy on Linux
- ⊘ Run the tests on Linux
- ⊘ Show debug info
- ⊘ Setup tmate session



osx-clang-clang13-cling
succeeded 13 hours ago in 3m 8s

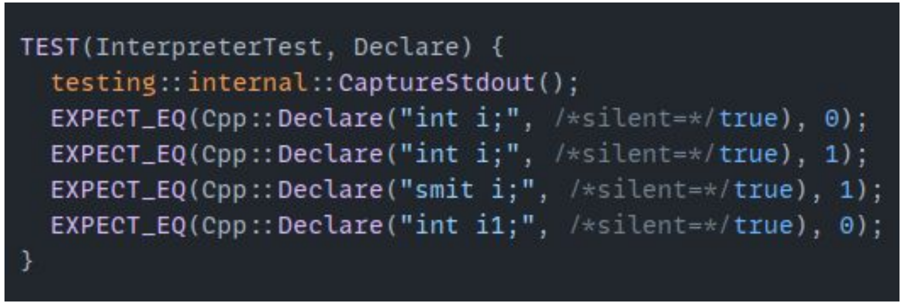
- ✓ Set up job
- ✓ Run actions/checkout@v3
- ✓ Set up Python
- ✓ Save PR Info
- ✓ Run nelonoel/branch-name@v1.0.1
- ✓ Setup default Build Type on *nux
- ⊘ Setup compiler on Linux
- ✓ Setup compiler on macOS
- ⊘ Install deps on Linux
- ✓ Install deps on MacOS
- ✓ Restore Cache LLVM/Clang runtime build directory
- ⊘ Build LLVM/Cling on Unix if the cache is invalid
- ⊘ Save Cache LLVM/Clang runtime build directory
- ⊘ Setup code coverage
- ✓ Build and Test/Install CppInterOp on Unix Systems
- ⊘ Build and Install cppyy-backend on Linux
- ⊘ Install CPyCppyy on Linux
- ⊘ Install cppyy on Linux
- ⊘ Run cppyy on Linux
- ⊘ Run the tests on Linux
- ⊘ Show debug info

End Phase

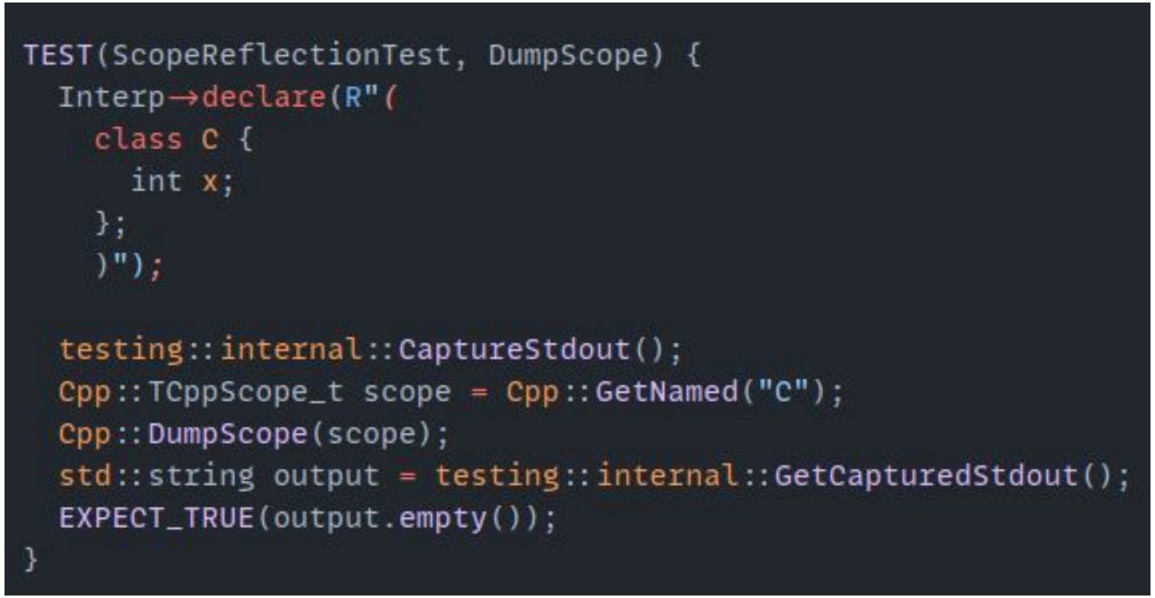
1] In the latter phase of the project, the focus shifted towards the implementation of Google Tests for specific modules within CppInterOp. Additionally, existing tests were modified to encompass all possible edge cases.

2] This meticulous testing approach is essential for ensuring the development of high-quality software that fulfills all its requirements comprehensively.

3] Writing these tests serves multiple purposes, such as gaining a deeper understanding of the module functionalities, identifying and eliminating potential bugs, and ultimately refining the software's overall reliability.



```cpp
TEST(InterpreterTest, Declare) {
  testing::internal::CaptureStdout();
  EXPECT_EQ(Cpp::Declare("int i;", /*silent=*/true), 0);
  EXPECT_EQ(Cpp::Declare("int i;", /*silent=*/true), 1);
  EXPECT_EQ(Cpp::Declare("smit i;", /*silent=*/true), 1);
  EXPECT_EQ(Cpp::Declare("int i1;", /*silent=*/true), 0);
}
```



```cpp
TEST(ScopeReflectionTest, DumpScope) {
  Interp->declare(R"(
    class C {
      int x;
    };
    )");

  testing::internal::CaptureStdout();
  Cpp::TCppScope_t scope = Cpp::GetNamed("C");
  Cpp::DumpScope(scope);
  std::string output = testing::internal::GetCapturedStdout();
  EXPECT_TRUE(output.empty());
}
```

Future Works

1] Integrated the Windows build structure into the GitHub Continuous Integration (CI) pipeline.

2] Adding tests for the remaining untested portions of the code to enhance code coverage.

3] Continuing exploration of cppyy and cppyy-backend for the purpose of debugging tests.

Learnings

1] I was completely new with compiler domain. However while implementing APIs during GSoC period. I was able to understand more about clang/cling. And this learning increased my curiosity in the field.

2] I also learnt a lot about build structure and Github CI and integration with it

3] I came across Google Test Framework and understood how it works and its requirement.

4] Most notably, this GSoC project marked my maiden voyage into collaborative project work, offering me a firsthand glimpse into what it's like to work on a team under someone else's guidance.

Acknowledgement

I am extremely grateful to my mentor **Vassil Vassilev** and **Baidyanath Kundu** for allowing me to work on the project and for all of their guidance and support. Whenever I sought assistance or guidance, they unfailingly stood by my side, offering their support and expertise.

Thank You