
C & C++ testing con Cest Framework

MeetUp C/C++ Madrid – Febrero 2025

¡Hola!

Gracias por estar hoy aquí.

¿Qué vamos a ver hoy?

¿Qué es Cest Framework?

¿Cuál es su historia?

¿Qué podemos hacer con él?

¡Demo!

Retos técnicos en su desarrollo

¿Qué hay planeado para el futuro?

¿Qué es Cest Framework?

Framework de testing para C++ y C

¿Qué es Cest Framework?

Framework de testing para C++ y C

Enfocado en la simplicidad y la legibilidad

¿Qué es Cest Framework?

Framework de testing para C++ y C

Enfocado en la simplicidad y la legibilidad

Fácil de usar, integrar y aprender

¿Qué es Cest Framework?

Framework de testing para C++ y C

Enfocado en la simplicidad y la legibilidad

Fácil de usar, integrar y aprender

Cargado de features

**¿Porqué crear otro
framework de testing?**

DevEx

Xtreme Programming

Anécdotas y experiencias

Pruebas con usuarios

¿Qué podemos hacer con Cest Framework?

Features: Descripción de los tests

El test más sencillo: una única suite

```
describe("hello world", []() {  
  it("prints hello!", []() {  
    std::cout << "hello!";  
  });  
  
  it("performs an assertion", []() {  
    expect("hello").toEqual("hello");  
  });  
});
```

describe()

it()

it()

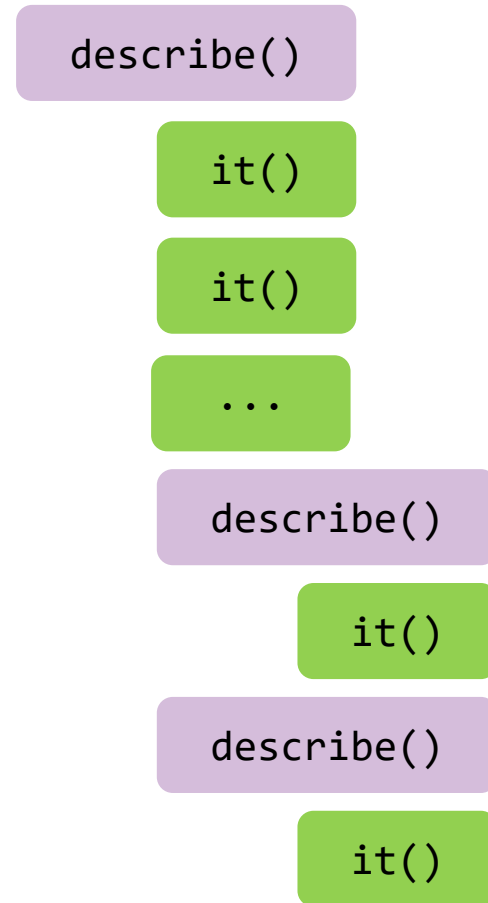
it()

...

Features: Descripción de los tests

Test suites anidadas: expresando dependencia o relación

```
describe("SomethingService", []() {  
  it("calculates", []() {});  
  it("re-calculates", []() {});  
  
  describe("when XYZ happens", []() {  
    it("does ABC", []() {});  
  });  
  
  describe("when there is an error", []() {  
    it("does not calculate", []() {});  
  });  
});
```



Features: Descripción de los tests

Test suites, tests, pre-condiciones y post-condiciones

```
Queue q;
```

```
describe("Queue", [&]() {  
    beforeAll([&]() { q.connect("address"); });  
  
    afterAll([]() { q.disconnect(); });  
  
    it("sends and receives messages", [&]() {  
        std::string message;  
        q.send(message);  
  
        const auto received = q.peak();  
  
        expect(received).toEqual(message);  
    });  
});
```

beforeAll()

it()

it()

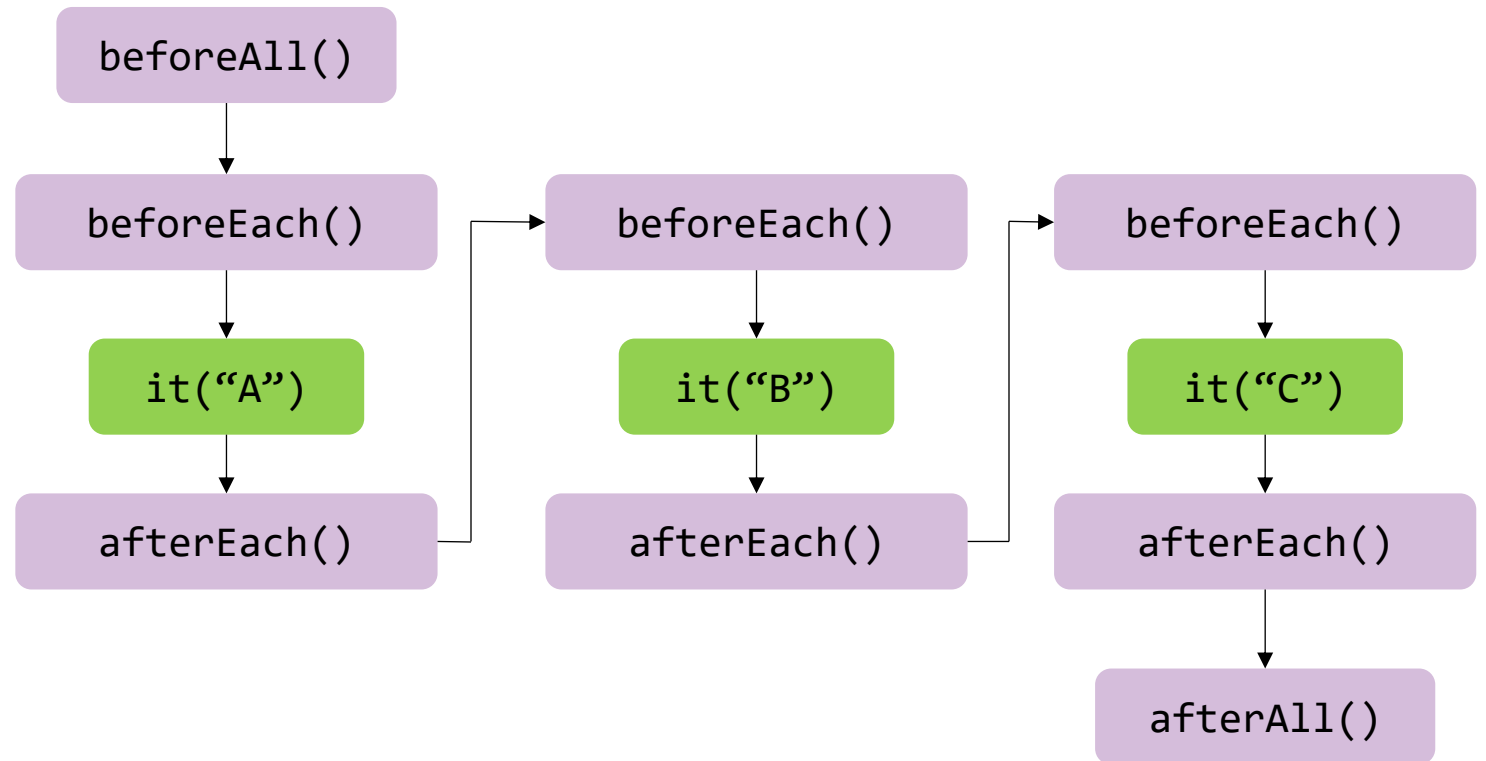
...

afterAll()

Features: Descripción de los tests

Test suites, tests, pre-condiciones y post-condiciones

```
describe("XYZ", []() {  
  beforeAll([]() {});  
  afterAll([]() {});  
  
  beforeEach([]() {});  
  afterEach([]() {});  
  
  it("A", []() {});  
  it("B", []() {});  
  it("C", []() {});  
});
```



Features: Descripción de los tests

Combinándolo todo...

```
describe("XYZ", []() {  
  afterAll([]() {});  
  afterEach([]() {});  
  it("A", []() {});  
  it("B", []() {});  
  
  describe("ABC", []() {  
    beforeEach([]() {});  
    beforeAll([]() {});  
    it("A", []() {});  
    it("B", []() {});  
  });  
});
```


Features: Matchers

Comparando enteros, flotantes, strings, arrays, punteros...

```
it("asserts equality", []() {  
    auto first = true;  
    auto second = 12345;  
  
    expect(first).toBe(true);  
    expect(second).toEqual(12345);  
    expect(first).Not->toBe(false);  
});
```

```
it("asserts strings", []() {  
    expect("hello").toBe("hello");  
    expect("world").toContain("rld");  
    expect("cest").toHaveLength(4);  
});
```

```
it("asserts RegEx matches", []() {  
    expect("Hello world cest").toMatch(Regex("^Hell.*cest$"));  
    expect("I have 12 apples").toMatch(Regex(".*\\d+ apples"));  
    expect("To match a partial match").toMatch(Regex("\\w match$"));  
});
```


Features: Matchers

Comparando enteros, flotantes, strings, arrays, punteros...

```
it("asserts equality", []() {  
    auto first = true;  
    auto second = 12345;  
  
    expect(first).toBe(true);  
    expect(second).toEqual(12345);  
    expect(first).Not->toBe(false);  
});
```

```
it("asserts strings", []() {  
    expect("hello").toBe("hello");  
    expect("world").toContain("rld");  
    expect("cest").toHaveLength(4);  
});
```

```
it("asserts RegEx matches", []() {  
    expect("Hello world cest").toMatch(Regex("^Hell.*cest$"));  
    expect("I have 12 apples").toMatch(Regex(".*\\d+ apples"));  
    expect("To match a partial match").toMatch(Regex("\\w match$"));  
});
```



std::regex

Features: Matchers

Comparando enteros, flotantes, strings, arrays, punteros...

```
it("asserts pointers", []() {  
    void *address = (void *)0xFA101132;  
    char *string = (char *)"something";  
    int year = 2019;  
    float *this_is_null = nullptr;  
  
    expect(address).toBe(  
        (void *)0xFA101132  
    );  
    expect(string).toEqualMemory(  
        (char *)"something",  
        strlen("something") + 1  
    );  
    expect(&year).toBeNotNull();  
    expect(this_is_null).toBeNull();  
});
```

```
it("asserts lists", []() {  
    std::vector<int> numbers({  
        10, 20, 30  
    });  
  
    expect(numbers).toContain(30);  
    expect(numbers).toBe(numbers);  
    expect(numbers).toHaveLength(3);  
});
```

Features: Matchers

Comparando enteros, flotantes, strings, arrays, punteros...

```
it("can have a custom ε provided", []() {  
    double d1 = 4.100,  
           d2 = 4.102,  
           epsilon = 0.01;  
  
    expect(d1).toEqual(d2, epsilon);  
    expect(d1).Not->toEqual(0, epsilon);  
});
```

```
it("supports arithmetic operations", []() {  
    expect(2.0f).toBeGreaterThan(1.0f);  
    expect(2.0f).toBeLessThan(3.0f);  
    expect(2.0).toBeGreaterThan(1.0);  
    expect(2.0).toBeLessThan(3.0);  
});
```

Features: Matchers

Capturando excepciones

```
using namespace std;
using namespace ctest;

describe("std::stoi", []() {
    it("only converts integers", []() {
        string number = "potato";
        stoi(number);
    });
});
```

```
ubuntu@tryit:~$ g++ -I. test.cpp -o test && ./test
FAIL test.cpp:7 only converts integers
Failed at line 7: Unhandled exception: stoi
  4 |
  5 | describe("std::stoi", []() {
  6 |     it("only converts integers", []() {
> 7 |         string number = "potato";
  8 |         stoi(number);
  9 |     });
 10 | });
ubuntu@tryit:~$
```

Features: Matchers

Capturando excepciones

```
using namespace std;
using namespace ctest;

describe("std::stoi", []() {
    it("only converts integers", []() {
        string number = "potato";

        assertRaises<invalid_argument>([&]() {
            stoi(number);
        });
    });
});
```

```
ubuntu@tryit:~$ g++ -I. test.cpp -o test && ./test
PASS test.cpp:7 only converts integers
ubuntu@tryit:~$
```

Features: Matchers

Capturando excepciones

```
using namespace std;
using namespace ctest;

describe("std::stoi", []() {
    it("only converts integers", []() {
        string number = "potato";

        assertRaises<invalid_argument>([=]() {
            //stoi(number);
        });
    });
});
```

```
ubuntu@tryit:~$ g++ -I. test.cpp -o test && ./test
FAIL test.cpp:7 only converts integers
Failed at line 7: Expected exception not raised
 4 |
 5 | describe("std::stoi", []() {
 6 |     it("only converts integers", []() {
> 7 |         string number = "potato";
 8 |
 9 |         assertRaises<invalid_argument>([=]() {
10 |             //stoi(number);
ubuntu@tryit:~$
```

Features: Tests parametrizados

Repitiendo el mismo test con múltiples valores de tipo T

```
using namespace cest;

describe("parametrized tests", []() {
    it("can make use of integer parameters", []() {
        withParameter<int>().
            withValue(10).
            withValue(20).
            withValue(30).
            thenDo([](int x) {
                expect(x > 0).toBeTruthy();
            });
    });
});
```


Features: Tests parametrizados

Repitiendo el mismo test con múltiples valores de tipo T

```
using namespace cest;
```

```
describe("parametrized tests", []() {  
    it("can make use of integer parameters", []() {  
        withParameter<int>().  
        withValue(10).  
        withValue(20).  
        withValue(30).  
        thenDo([](int x) {  
            expect(x < 20).toBeTruthy();  
        });  
    });  
});
```

```
ubuntu@tryit:~$ g++ -I. test.cpp -o test && ./test  
FAIL test.cpp:7 can make use of integer parameters  
Failed at line 12: Expression is not truthy  
  9 |           withValue(20).  
 10 |           withValue(30).  
 11 |           thenDo([](int x) {  
> 12 |               expect(x < 20).toBeTruthy();  
 13 |           });  
 14 |       });  
 15 |   });  
ubuntu@tryit:~$
```

Features: Detección automática de leaks

Incorpora soporte de Address Sanitizer

```
FAIL leak.test.cpp:4 does not free its memory
Failed at line 4: Detected potential memory leaks during test execution.
1 |
2 | describe("Test with leaks", []() {
3 |     it("does not free its memory", []() {
> 4 |         int *ptr = new int;
5 |         *ptr = 123;
6 |     });
7 | });
AddressSanitizer result:

=====
==14523==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 4 byte(s) in 1 object(s) allocated from:
#0 0x7fb10bd82647 in operator new(unsigned long) ../../../../src/libsanitizer/asan/asan_new_delete.cpp:99
#1 0x55aelc44574f in operator() /tmp/leak.test.cpp:3
#2 0x55aelc4473a9 in __invoke_impl<void, <lambda()>::<lambda()>&& /usr/include/c++/10/bits/invoke.h:60
#3 0x55aelc446dfc in __invoke_r<void, <lambda()>::<lambda()>&& /usr/include/c++/10/bits/invoke.h:153
#4 0x55aelc446696 in _M_invoke /usr/include/c++/10/bits/std_function.h:291
#5 0x55aelc44aa07 in std::function<void ()>::operator()() const /usr/include/c++/10/bits/std_function.h:622
#6 0x55aelc442584 in ctest::runTestSuite(ctest::TestSuite*) ctest:664
#7 0x55aelc445390 in main ctest:1366
#8 0x7fb10b93fd09 in __libc_start_main ../csu/libc-start.c:308

SUMMARY: AddressSanitizer: 4 byte(s) leaked in 1 allocation(s).
```

Features: Runner de tests

Herramienta para ejecutar tests tanto en local como en CI

```
x should greet shouting when name is uppercase

Failed at line 98: Expected "HEEEELLO PEPE!!!!", was "HELLO PEPE!!!!"

95 |
96 |     auto greeting = Greet(name);
97 |
> 98 |     expect(greeting).toBe("HEEEELLO PEPE!!!!");
99 | });
100 |
101 | it("should greet my friend when greeting an empty list of names", []() {

FAIL /home/cesar/github/cest/test/framework/command-line-arguments.test.cpp

x will fail to use provided seed for randomized tests if seed is not present

Failed at line 84: Expected 1, was 0

81 |
82 |     options = cest::parseArgs(argc, argv);
83 |
> 84 |     expect(options.random_seed_present).toBe(true);
85 | });
86 |
87 | it("will fail to use provided seed for randomized tests if seed is not an integer", []() {
}) {

Test Suites: 2 failed, 2 total
Tests:       2 failed, 15 passed, 17 total
Time:        0.004196 s
Ran all test suites.

Watch Usage
> Press f to run only failed tests.
> Press p to filter by a filename regex pattern.
> Press q to quit watch mode.
> Press Enter to trigger a test run.
```

```
Input the test .cpp file filter to run.
Press Enter to submit:
> greet

Found matches:
/home/cesar/github/cest/test/examples/greeting.test.cpp
```

```
greeting kata
✓ should greet my friend when name is empty
✓ should greet
x should greet shouting when name is uppercase

Failed at line 98: Expected "HEEEELLO PEPE!!!!", was "HELLO PEPE!!!!"
95 |
96 |     auto greeting = Greet(name);
97 |
> 98 |     expect(greeting).toBe("HEEEELLO PEPE!!!!");
99 | });
100 |
101 | it("should greet my friend when greeting an empty list of names", []() {

✓ should greet my friend when greeting an empty list of names
✓ should greet a list of names with one name
✓ should greet two people
✓ should greet many people


Test Suites: 1 failed, 1 total
Tests:       1 failed, 6 passed, 7 total
Time:        0.001923 s
Ran all test suites.


Watch Usage
> Press f to run only failed tests.
> Press p to filter by a filename regex pattern.
> Press q to quit watch mode.
> Press Enter to trigger a test run.
```

Features: Runner de tests

Herramienta para ejecutar tests tanto en local como en CI

```
85 ./build/cest-runner
86 PASS /home/runner/work/cest/cest/test/examples/chatbot.test.cpp
87 PASS /home/runner/work/cest/cest/runner/test/cmd-args.test.cpp
88 PASS /home/runner/work/cest/cest/test/framework/empty.test.cpp
89 PASS /home/runner/work/cest/cest/test/framework/randomized-tests.test.cpp
90 PASS /home/runner/work/cest/cest/test/examples/roman-arabic-converter.test.cpp
91 PASS /home/runner/work/cest/cest/test/framework/nested-suites.test.cpp
92 PASS /home/runner/work/cest/cest/test/framework/parametrized-tests.test.cpp
93 PASS /home/runner/work/cest/cest/test/framework/custom-assertions.test.cpp
94 PASS /home/runner/work/cest/cest/test/examples/money.test.cpp
95 PASS /home/runner/work/cest/cest/test/framework/floating_point.test.cpp
96 PASS /home/runner/work/cest/cest/test/examples/greeting.test.cpp
97 PASS /home/runner/work/cest/cest/runner/test/runner.test.cpp
98 PASS /home/runner/work/cest/cest/test/framework/assertions.test.cpp
99 PASS /home/runner/work/cest/cest/test/framework/command-line-arguments.test.cpp
100 PASS /home/runner/work/cest/cest/test/framework/description.test.cpp
101
102 Test Suites: 15 passed 15 total
103 Tests:      69 passed 1 skipped 70 total
104 Time:       0.277817 s
105 Ran all test suites.
```

>  Post Run actions/checkout@master

>  Complete job

¡Demo!

Vamos a probar Cest Framework

Retos Técnicos

Single-header Library

Test tree Discovery

Memory Leak Detection

Mejoras técnicas pendientes

Retos Técnicos: Single-header library

¿Qué pros y contras tiene usar una single-header library?

- ✓ Fácil de usar (un sólo `#include` y listo)
- ✓ Compilación muy simple
- ✓ Fácil de instalar → un único fichero

- ✗ Gestión del ciclo de desarrollo complejo
- ✗ Tiempos de compilación más lentos
- ✗ Translation Units más grandes

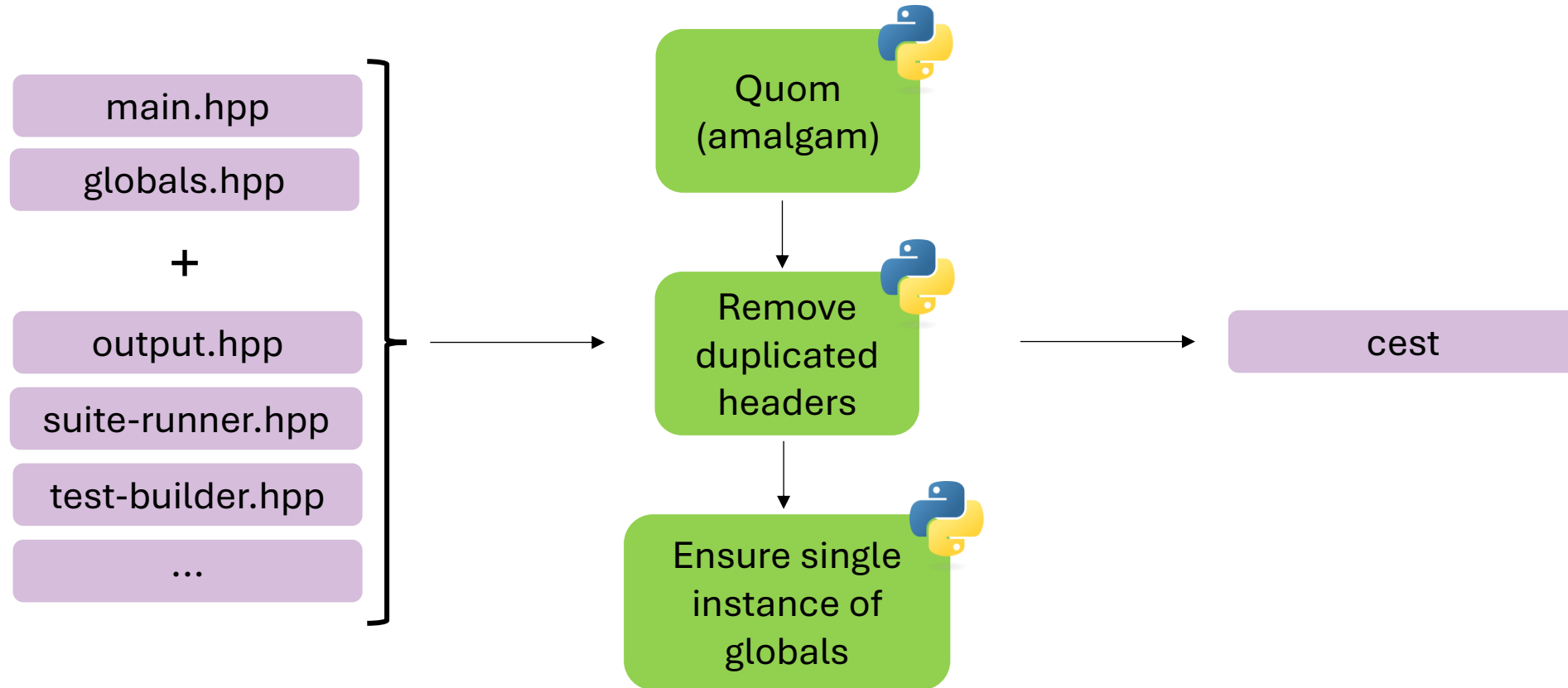
Retos Técnicos: Single-header library

¿Cuáles han sido los principales retos?

- Evitar duplicación de cabeceras (tamaño de la TU)
- Dividir el código en módulos de forma efectiva
- Gestionar el estado global desde los distintos módulos

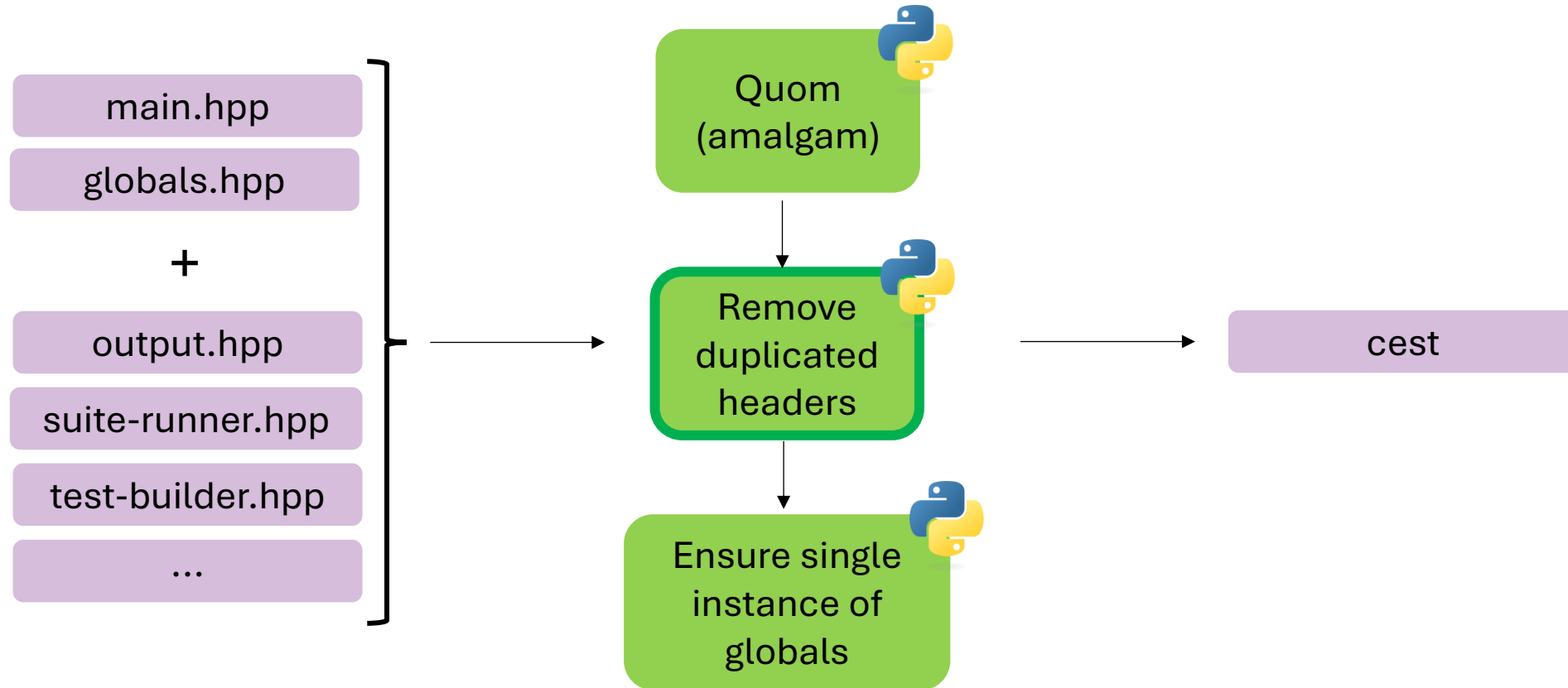
Retos Técnicos: Single-header library

Proceso de compilación de la cabecera



Retos Técnicos: Single-header library

Proceso de compilación de la cabecera



Retos Técnicos: Single-header library

Borrado de #includes duplicados

parametrized.hpp

```
#pragma once
#include <vector>
#include <functional>

namespace cest {
    template <class T>
    class Parameter {
        // ...
    }
}
```

expect.hpp

```
#pragma once
#include <sstream>
#include <regex>
#include <cmath>
#include "types.hpp"

// ...
```

types.hpp

```
#pragma once
#include <functional>
#include <map>
#include <string>
#include <csetjmp>

namespace cest {
    // ...
}
```

Retos Técnicos: Single-header library

Borrado de #includes duplicados

main.hpp

```
#include "types.hpp"  
#include "parametrized.hpp"  
#include "expect.hpp"  
// ...
```

cest (raw)

```
#include <vector>  
#include <functional>  
// ...  
#include <sstream>  
#include <regex>  
#include <cmath>  
// ...  
#include <functional>  
#include <map>  
#include <string>  
#include <csetjmp>  
// ...
```

Retos Técnicos: Single-header library

Borrado de #includes duplicados

main.hpp

```
#include "types.hpp"  
#include "parametrized.hpp"  
#include "expect.hpp"  
// ...
```

cest (raw)

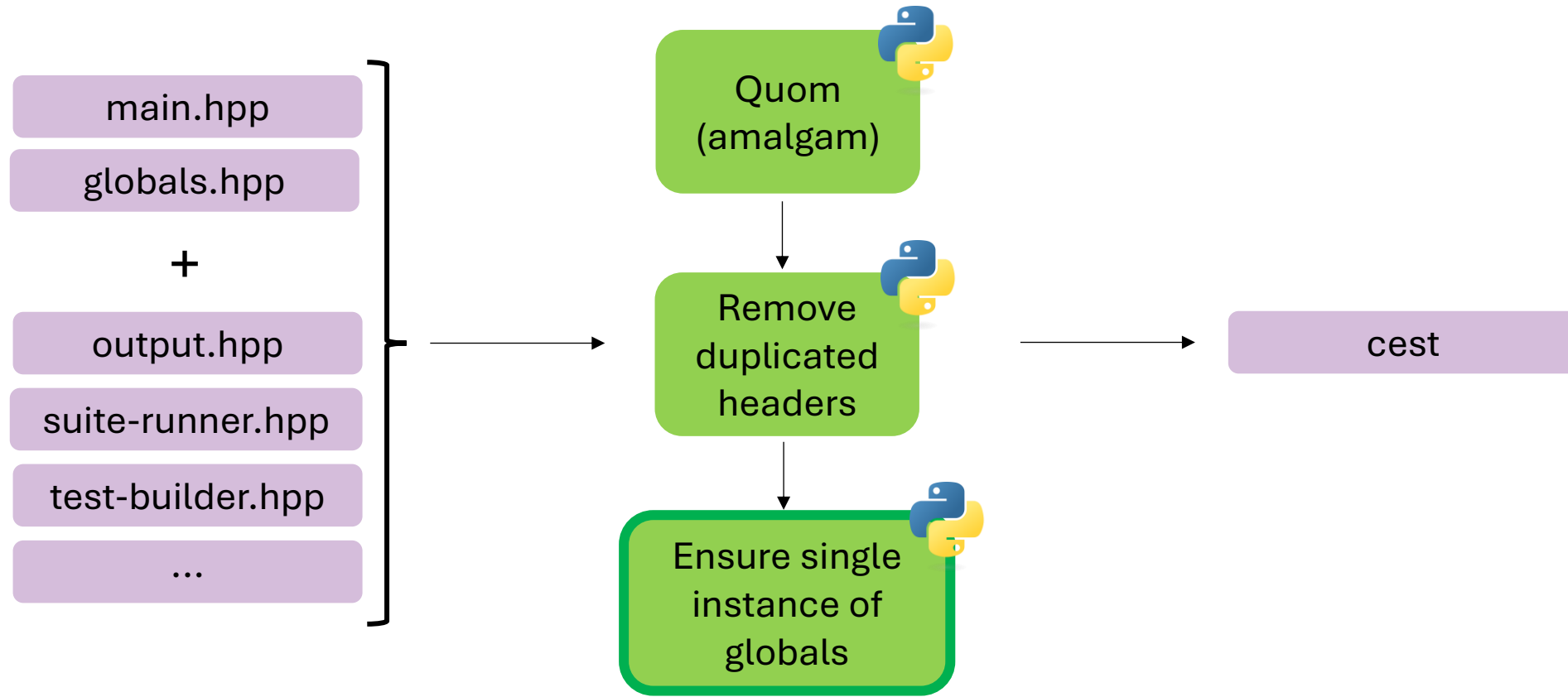
```
#include <vector>  
#include <functional>  
// ...  
#include <sstream>  
#include <regex>  
#include <cmath>  
// ...  
#include <functional>  
#include <map>  
#include <string>  
#include <csetjmp>  
// ...
```

cest

```
#include <vector>  
#include <sstream>  
#include <regex>  
#include <cmath>  
#include <functional>  
#include <map>  
#include <string>  
#include <csetjmp>  
// ...
```

Retos Técnicos: Single-header library

Proceso de compilación de la cabecera



Retos Técnicos: Single-header library

Asegurando una única instancia de las globales

globals.hpp

```
#pragma once
#include "types.hpp"

// CEST-ONCE-START
static cest::CestGlobals __cest_globals;
// CEST-ONCE-END
```

main.hpp

```
#include "types.hpp"
#include "globals.hpp"
#include "output.hpp"
#include "...

int main(int argc, const char *argv[]) {
    cest::TestSuite *root_suite =
        &__cest_globals.root_test_suite;

    cest::CommandLineOptions
        command_line_options =
            cest::parseArgs(argc, argv);

    // ...
```

Retos Técnicos: Single-header library

Asegurando una única instancia de las globales

globals.hpp

```
#pragma once
#include "types.hpp"

// CEST-ONCE-START
static cest::CestGlobals __cest_globals;
// CEST-ONCE-END
```

main.hpp

```
#include "types.hpp"
#include "globals.hpp"
#include "output.hpp"
#include "...

int main(int argc, const char *argv[]) {
    cest::TestSuite *root_suite =
        &__cest_globals.root_test_suite;

    cest::CommandLineOptions
        command_line_options =
            cest::parseArgs(argc, argv);

    // ...
```


Retos Técnicos: Test Tree Discovery

Ejemplo de test de Cest Framework

```
#include <cest>

describe("DatabaseConnection", []() {
    it("connects to a Postgres DB", []() {});
    it("connects to an Oracle DB", []() {});

    describe("when connection succeeds", []() {
        it("can perform queries", []() {});
    });

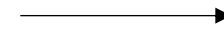
    describe("when connection fails", []() {
        it("DatabaseException is thrown", []() {});
    });
});
```

Retos Técnicos: Test Tree Discovery

Simplificando los nombres

```
#include <ctest>
```

```
describe("DatabaseConnection", []() {  
    it("connects to a Postgres DB", []() {});  
    it("connects to an Oracle DB", []() {});  
  
    describe("when connection succeeds", []() {  
        it("can perform queries", []() {});  
    });  
  
    describe("when connection fails", []() {  
        it("DatabaseException is thrown", []() {});  
    });  
});
```



```
#include <ctest>
```

```
describe("s1", []() {  
    it("T1", []() {});  
    it("T2", []() {});  
  
    describe("s2", []() {  
        it("T3", []() {});  
    });  
  
    describe("s3", []() {  
        it("T4", []() {});  
    });  
});
```

Retos Técnicos: Test Tree Discovery

Unrolling de las macros

```
#include <ctest>
```

```
__attribute__((unused)) static int dummy3 = ctest::describeFn("S1", 3, "test.cpp", []() {  
    ctest::itFn("T1", 4, "test.cpp", []() {});  
    ctest::itFn("T2", 5, "test.cpp", []() {});  
  
    __attribute__((unused)) static int dummy7 = ctest::describeFn("S2", 7, "test.cpp", []() {  
        ctest::itFn("T3", 8, "test.cpp", []() {});  
    });  
  
    __attribute__((unused)) static int dummy11 = ctest::describeFn("S3", 11, "test.cpp", []() {  
        ctest::itFn("T4", 12, "test.cpp", []() {});  
    });  
});
```

Retos Técnicos: Test Tree Discovery

Unrolling de las macros

```
#include <ctest>
```

```
__attribute__((unused)) static int dummy3 = ctest::describeFn("S1", 3, "test.cpp", []() {  
    ctest::itFn("T1", 4, "test.cpp", []() {});  
    ctest::itFn("T2", 5, "test.cpp", []() {});  
  
    __attribute__((unused)) static int dummy7 = ctest::describeFn("S2", 7, "test.cpp", []() {  
        ctest::itFn("T3", 8, "test.cpp", []() {});  
    });  
  
    __attribute__((unused)) static int dummy11 = ctest::describeFn("S3", 11, "test.cpp", []() {  
        ctest::itFn("T4", 12, "test.cpp", []() {});  
    });  
});
```

Retos Técnicos: Test Tree Discovery

Construyendo el árbol de tests

```
#include <ctest>
```

```
describe("s1", []() {  
    it("T1", []() {});  
    it("T2", []() {});  
  
    describe("s2", []() {  
        it("T3", []() {});  
    });  
  
    describe("s3", []() {  
        it("T4", []() {});  
    });  
});
```

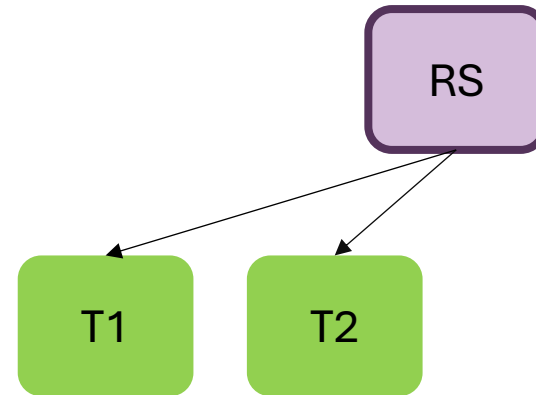
RS

Retos Técnicos: Test Tree Discovery

Construyendo el árbol de tests

```
#include <ctest>
```

```
describe("s1", []() {  
    it("T1", []() {});  
    it("T2", []() {});  
  
    describe("s2", []() {  
        it("T3", []() {});  
    });  
  
    describe("s3", []() {  
        it("T4", []() {});  
    });  
});
```



Retos Técnicos: Test Tree Discovery

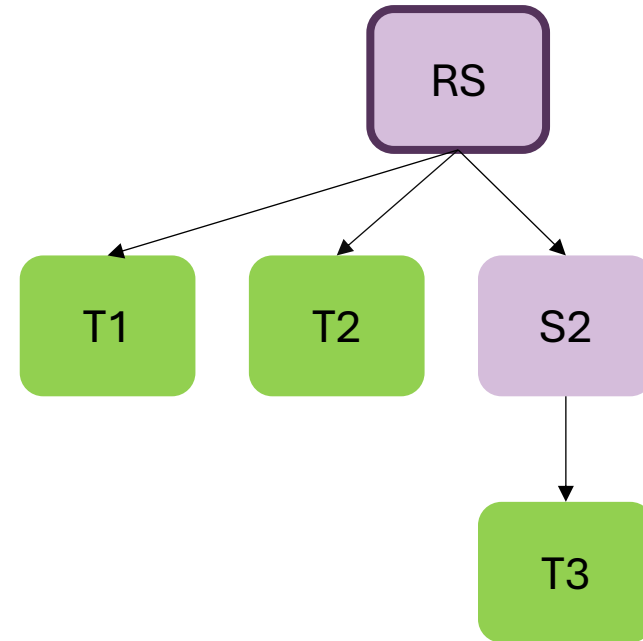
Construyendo el árbol de tests

```
#include <ctest>
```

```
describe("s1", []() {  
    it("T1", []() {});  
    it("T2", []() {});  
});
```

```
describe("s2", []() {  
    it("T3", []() {});  
});
```

```
describe("s3", []() {  
    it("T4", []() {});  
});  
});
```



Retos Técnicos: Test Tree Discovery

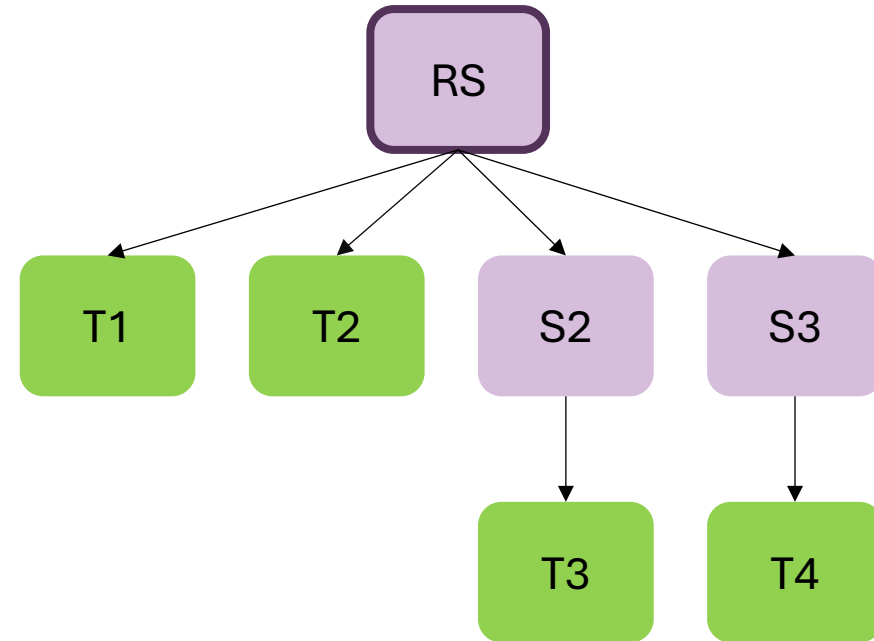
Construyendo el árbol de tests

```
#include <ctest>
```

```
describe("s1", []() {  
    it("T1", []() {});  
    it("T2", []() {});  
});
```

```
describe("s2", []() {  
    it("T3", []() {});  
});
```

```
describe("s3", []() {  
    it("T4", []() {});  
});  
});
```



Retos Técnicos: Test Tree Discovery

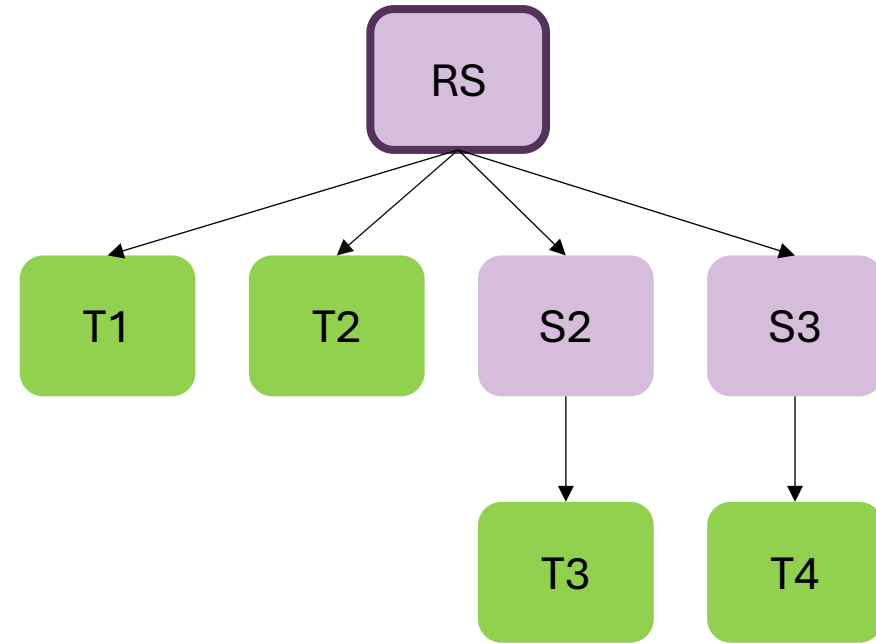
Construyendo el árbol de tests

```
#include <ctest>

describe("s1", []() {
    it("T1", []() {});
    it("T2", []() {});

    describe("s2", []() {
        it("T3", []() {});
    });

    describe("s3", []() {
        it("T4", []() {});
    });
});
```



Retos Técnicos: Test Tree Discovery

Pros y cons de esta estructura

- | | |
|--|--|
| ✓ Descripción de tests expresiva y familiar a otros lenguajes | ✗ Dependencia de macros, complicando el código |
| ✓ Fácil expresar dependencias y relaciones entre comportamientos | ✗ Problemas con la depuración (mejor con compiladores recientes) |
| ✓ Aumenta la legibilidad | ✗ Errores de compilación muy verbosos por la macro expansion |

Retos Técnicos: Memory Leak Detection

¿Cómo podemos detectar leaks?

- Overrides de `malloc` / `realloc` / `free` / `operator new` / `new[]`
- Wrapping de los ejecutables del test con Valgrind
- Compilar con ASAN e invocarlo en tiempo de test

Retos Técnicos: Memory Leak Detection

¿Cómo podemos detectar leaks?

- ✗ Overrides de `malloc` / `realloc` / `free` / `operator new` / `new[]` →
 - GCC `-Wl, malloc-override` funciona bien con la libc
 - ¿Qué pasa con C++?
 - Sobrecarga `new` / `new[]`
 - ¿Allocations internas?
 - ¿`make_unique` y cia?
- Wrapping de los ejecutables del test con Valgrind
- Compilar con ASAN e invocarlo en tiempo de test




Retos Técnicos: Memory Leak Detection

¿Cómo podemos detectar leaks?

- ❌ Overrides de `malloc` / `realloc` / `free` / `operator new` / `new[]`
 - No funciona a no ser que se use ctest-runner
- ❌ Wrapping de los ejecutables del test —————→ con Valgrind
 - Obliga a tener una distribución de Valgrind instalada al usuario
- Compilar con ASAN e invocarlo en tiempo de test

Retos Técnicos: Memory Leak Detection

¿Cómo podemos detectar leaks?

-  Overrides de `malloc` / `realloc` / `free` / `operator new` / `new[]`
 - Soportado en GCC y Clang modernos
-  Wrapping de los ejecutables del test con Valgrind
 - Cest se puede utilizar de forma stand-alone
-  Compilar con ASAN e invocarlo en tiempo de test →
 - No requiere instalación por parte del usuario

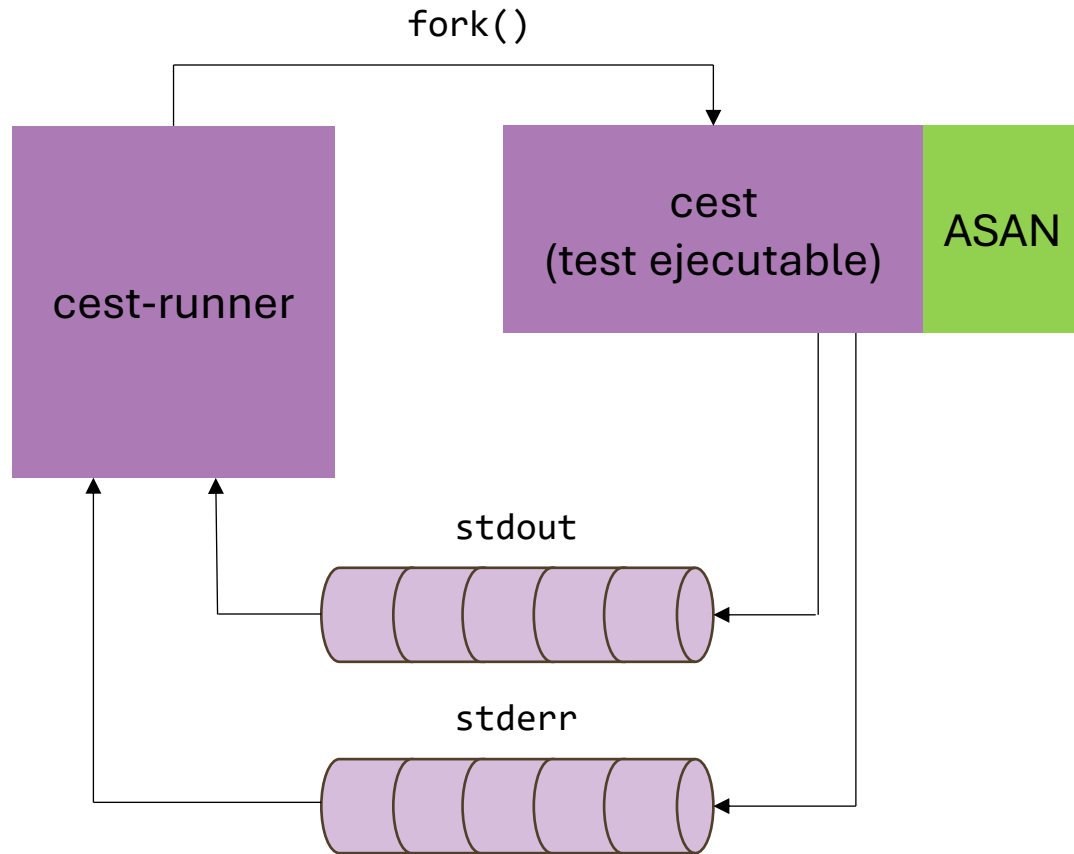
Retos Técnicos: Memory Leak Detection

Utilizando ASAN para detectar memory leaks

- API C para detección en runtime:
`__lsan_do_recoverable_leak_check()`
- Reporta != 0 si se ha producido un memory leak
- No hay API para obtener los resultados → sólo por stderr

Retos Técnicos: Memory Leak Detection

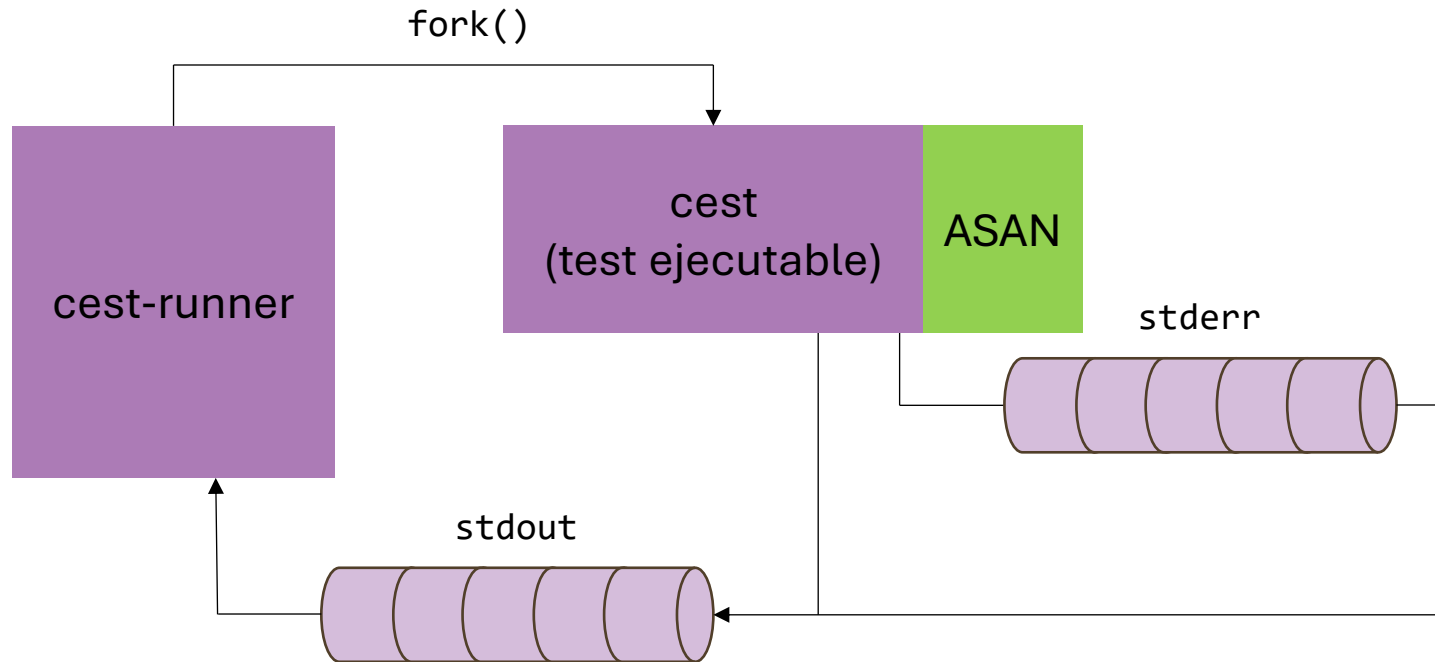
Leyendo stderr para parsear ASAN



- ✗ Los tests deben seguir siendo ejecutables stand-alone
- ✗ El código bajo test puede usar stderr por su cuenta → hay que cerrar el stream durante el test

Retos Técnicos: Memory Leak Detection

Leyendo stderr del propio proceso con pipes



- Es posible mutear stderr durante la ejecución del test (guardando el FD)
- Al acabar, ASAN emite el resultado por stderr → se redirige a stdout

Retos Técnicos: Mejoras pendientes

Ideas para mejorar el código de Cest Framework

- Utilizar `std::print` de C++23 para sustituir printing mediante streams
- Duplicación en el código de los matchers (`expect()`)
 - Se puede mejorar con un mejor uso de la especialización de templates
- Mejorar los matchers para facilitar hacer `expect<T>()` de cualquier tipo T
 - Ahora hay que hacer custom matchers complejos, o sobrecargar streams y operadores en el tipo a comparar

¿Qué hay para el futuro?

Mejores Matchers

Detección automática recompilación en runner

Tests basados en propiedades

Soporte para Windows

Futuro: Mejores Matchers

Matching de genéricos, colecciones built-in

```
it("compares any T object", [])() {  
    Transaction t1(123), t2(456);  
  
    expect(t1).toEqual(t2);  
});
```

```
it("matches arrays, maps...", [])() {  
    std::map<std::string, int> m;  
    std::array<char, 32> a;  
  
    expect(m).toHaveKey("hello");  
    expect(m).toContain(123);  
    expect(a).toContain('c');  
});
```

Futuro: Property-based tests

Testeando contra un conjunto de datos aleatorios

```
it("can generate numeric properties", []() {  
    withProperty<int>()  
        .minValue(0)  
        .maxValue(1000)  
        .thenDo([](int x) {  
            expect(x < 1000).toBeTruthy();  
        });  
});
```

```
it("can generate custom properties", []() {  
    withProperty<Vector3>()  
        .withGenerator(generateNegativeVec3)  
        .thenDo([](Vector3 vec) {  
            expect(vec.x < 0).toBeTruthy();  
            expect(vec.y < 0).toBeTruthy();  
            expect(vec.z < 0).toBeTruthy();  
            expect(vec.magnitude()).toBeGreaterThan(0.0);  
        });  
});
```

En conclusión...

Continuemos mejorando el ecosistema

En conclusión...

Continuemos mejorando el ecosistema

Mejor experiencia → mejores productos

En conclusión...

Continuemos mejorando el ecosistema

Mejor experiencia → mejores productos

Menos barrera de entrada → más innovación

**¡Muchísimas
gracias!**