
Core C++ 2024
Mastering CTest
Alex Kushnir

Mastering CTest

Alex Kushnir
Software engineer, Johnson&Johnson MedTech
28-Nov-2024
Core C++ 2024 conference, Tel-Aviv

Johnson&Johnson
MedTech

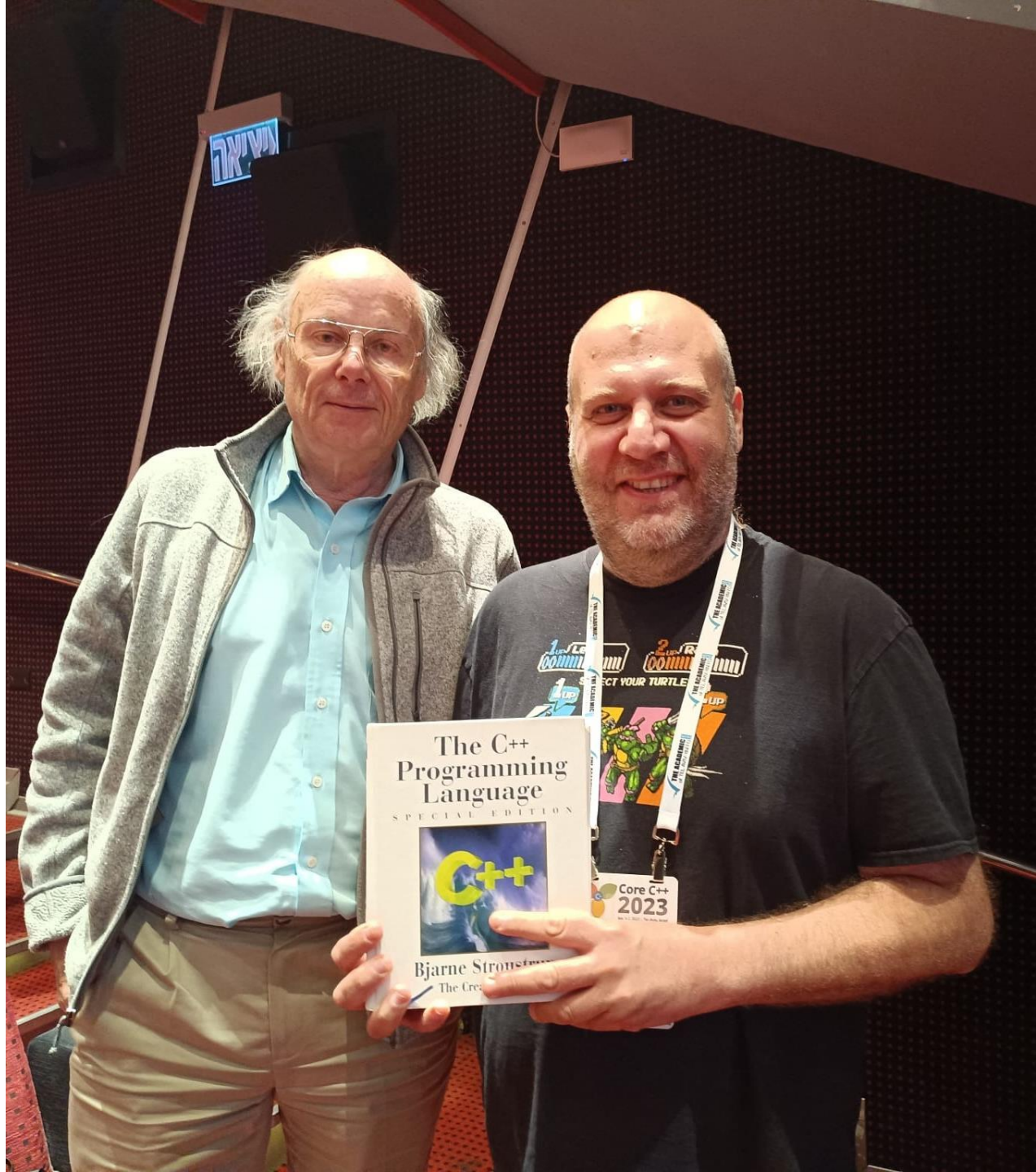
Electrophysiology

Streamline your testing workflow

CTest, part of the CMake suite offers a robust framework for automating and managing tests

About me

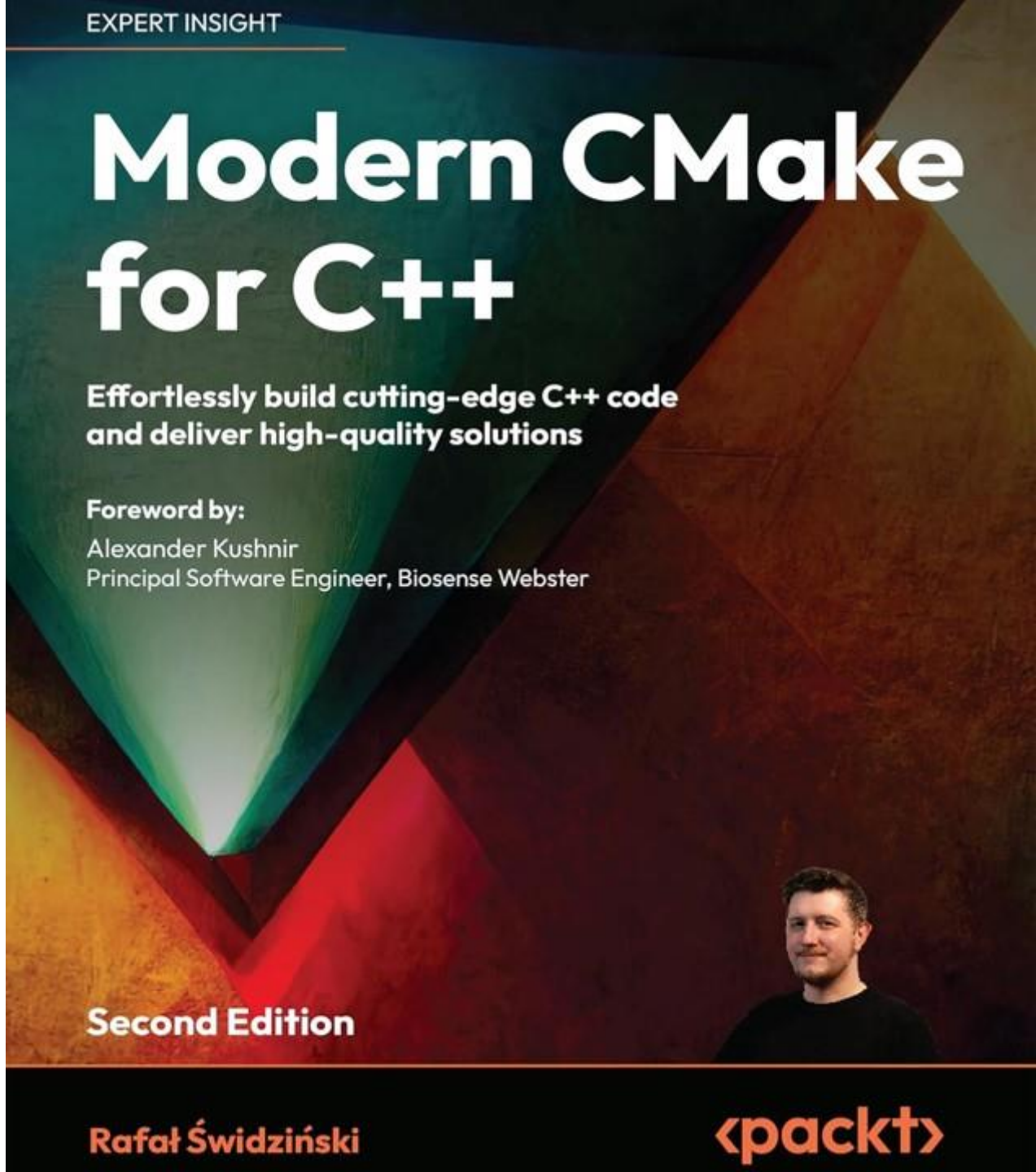
- Software engineer since 2007
- Mostly in embedded and low-level domains
- C++ = ♡
- Focusing on methodologies and tools



Modern CMake for C++

- Wrote a foreword for the book
- Reached out via LinkedIn
- Amazon bestseller

Best Sellers Rank: #131,969 in Books (See Top 100 in Books)
#8 in Software Programming Compilers
#18 in C++ Programming Language
#24 in Software Design Tools



J&J MedTech - Electrophysiology

- A global leader in diagnosing and treating complex arrhythmias
- Our mission is to cure AFib
- We are a team of professionals within wide spectrum of domains
- Software and hardware engineers, physicians, clinical specialists and many more

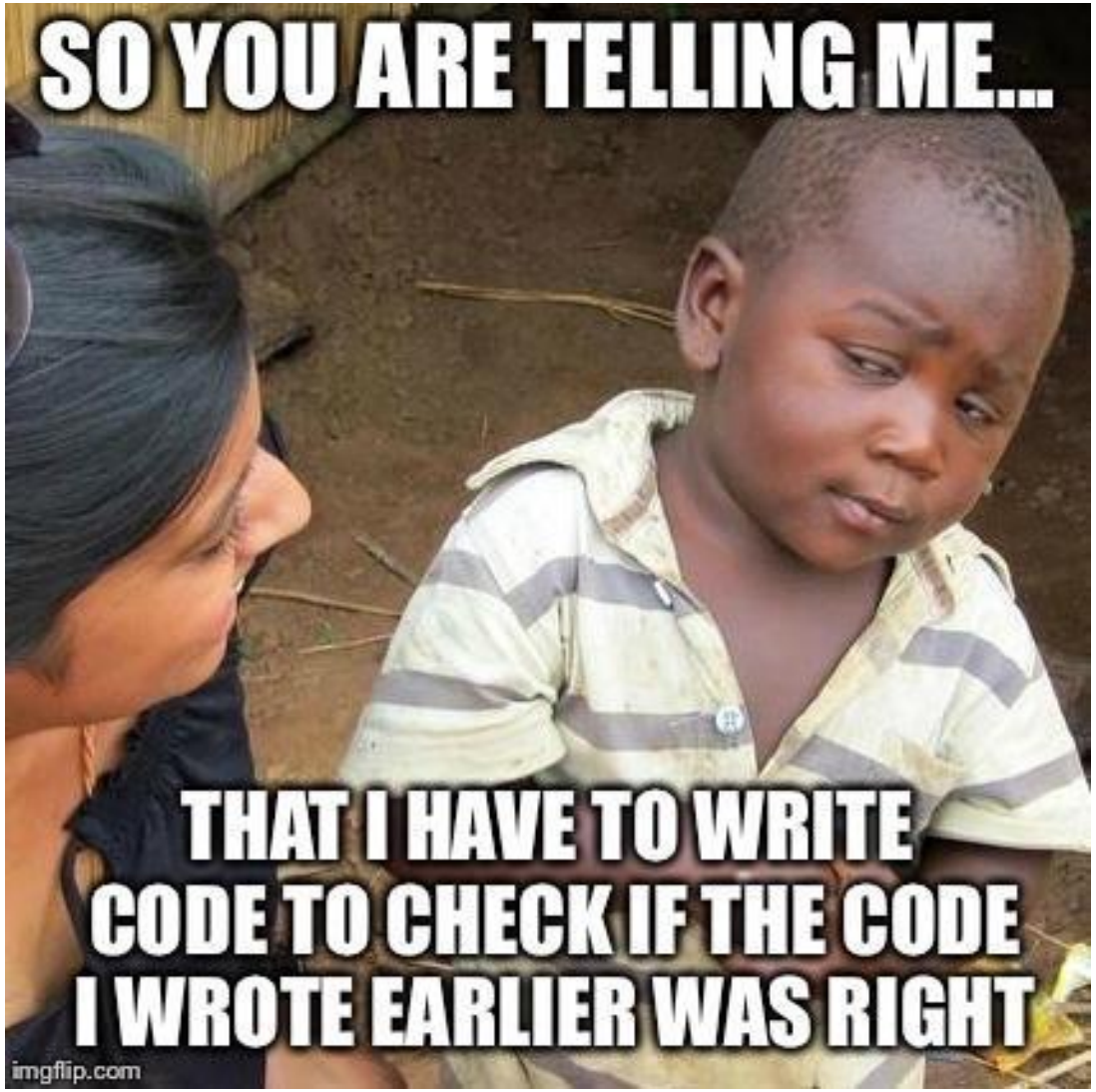
Our software team

- ~70 engineers
- ~4.5M LOC
- Modern C++
- C# for UI
- Windows, embedded devices

Agenda

- Unit testing – build trust
- CTest - your testing ally
- Setting up a basic test project
- Test coverage
- Q&A

What is unit testing?



Unit testing – build trust

- Catch bugs early
- Consistency and reliability
- Faster development cycles
- Encourages better code design
- Test as documentation – API usage

"I am not a great programmer; I am a good programmer with great habits"

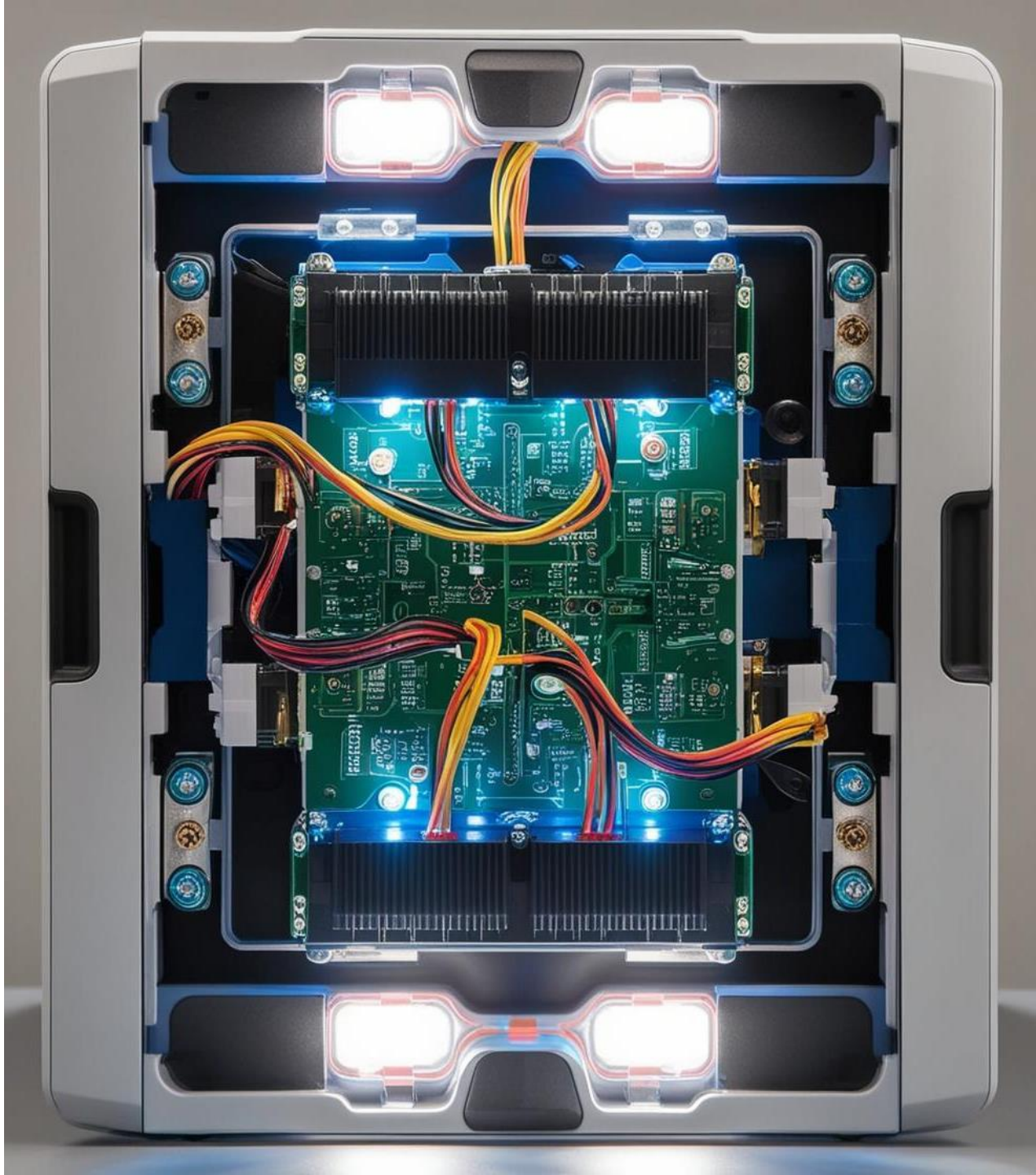
Kent Beck, creator of extreme programming

CTest: your testing ally

- Seamless integration with CMake
- Automation of testing during development
- Designed to be agnostic to test framework
- Dashboard integration with CDash

Setting up a basic project

- **System under test**
  - A class that represents a log message sent from embedded device to host
  - Limited length
  - Format string with arbitrary number of arguments (printf-like)
  - (“Test string with % of args”, num_args)
- **Possible test cases**
  - Happy path
  - Unmatched keys and arguments
  - Empty string
  - Text truncation



Unit under test

```cpp
template<class Tuple, std::size_t N>
struct TuplePrinter
{
    static void Print(const std::string& _format, std::ostream& _os,
        const Tuple& _t)
    {
        const size_t idx = _format.find_last_of(VARIABLE_KEY);
        TuplePrinter<Tuple, N - 1>::Print(
            std::string(_format, 0, idx), _os, _t);
        _os << std::get<N - 1>(_t) << std::string(_format, idx + 1);
    }
};

template<class Tuple>
struct TuplePrinter<Tuple, 1>
{
    static void Print(const std::string& _format, std::ostream& _os,
        const Tuple& _t)
    {
        const size_t idx = _format.find_first_of(VARIABLE_KEY);
        _os << std::string(_format, 0, idx) << std::get<0>(_t) <<
            std::string(_format, idx + 1);
    }
};
```

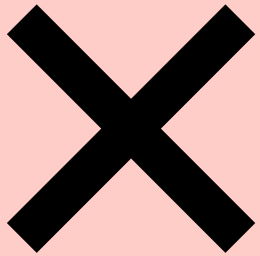
```cpp
template<class... Args>
std::string Format(const std::string& _format, Args&&... _args)
{
    std::stringstream ss;

    if constexpr (sizeof...(_args) == 0)
    {
        ss << _format;
    }
    else
    {
        if (IsFormatValid(_format, sizeof...(_args)))
        {
            if (_format.find_last_of(VARIABLE_KEY) != _format.npos)
            {
                const auto t = std::make_tuple(
                    std::forward<Args>(_args)...);
                TuplePrinter<
                    decltype(t), sizeof...(Args)>::Print(_format,
                    ss, t);
            }
        }
    }

    return ss.str();
}
```

Tests implementation

Implemented using 3 different methods, just for fun

Without framework

A bunch of functions. The decision which test to run is controlled via command line parameter



googletest

Test framework by google. Controlled by googletest built-in options and CMake integration

Catch2 framework

Same concept as googletest, different syntax

Tests implementation

googletest
Google C++ Testing Framework

Catch2

```
static void RemoteMessageTest_NoArguments()
{
    const std::string testString{
        "This is a test message without arguments" };

    auto rlm = CreateLogMessage(testString);
    VerifyMetaData(rlm);

    if (rlm.GetLogText() != testString)
    {
        std::exit(1);
    }

}

static void RemoteMessageTest_EmptyMessage()
{
    const std::string testString{ "" };

    auto rlm = CreateLogMessage(testString);
    VerifyMetaData(rlm);

    if (rlm.GetLogMessageLength() != testString.size() ||
        rlm.GetLogText() != testString)
    {
        std::exit(1);
    }
}
```

```
TEST_F(RemoteMessageTest, gtest_TextTruncation)
{
    // 104 Symbols
    const std::string testString{
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
        "abcdefghijklmnopqrstuvwxyz"
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
        "abcdefghijklmnopqrstuvwxyz" };
    const std::string expectedResult{ testString.begin(),
        testString.begin() +
        RemoteLogMessage::GetMaxLogMessageLength() };

    auto rlm = CreateLogMessage(testString);
    VerifyMetaData(rlm);

    ASSERT_EQ(rlm.GetLogMessageLength(), expectedResult.size());
}

TEST_F(RemoteMessageTest, gtest_VariadicArguments)
{
    std::tuple<std::int32_t, std::string, char> testTuple =
        std::make_tuple(123, "A string", 'x');
    std::stringstream expectedResultStream;
    expectedResultStream << "A " << std::get<0>(testTuple) <<
        " variadic " <<  std::get<1>(testTuple) << " message " <<
        std::get<2>(testTuple) <<  " for test";

    auto rlm = CreateLogMessage("A % variadic % message % for test",
        std::get<0>(testTuple), std::get<1>(testTuple),
        std::get<2>(testTuple));

    VerifyMetaData(rlm);

    ASSERT_EQ(expectedResultStream.str(), rlm.GetLogText());
}
```

```
TEST_CASE_METHOD(RemoteMessageTest,
    "catch2_NoKeyForArgs", "[RMsg]")
{
    std::tuple<std::int32_t, std::string, char> testTuple =
        std::make_tuple(123, "A string", 'x');
    std::string format{ "A format without percents" };
    auto rlm = CreateLogMessage(format, std::get<0>(testTuple),
        std::get<1>(testTuple), std::get<2>(testTuple));
    CHECK(std::string{ rlm.GetLogText() } == std::string{});
}

TEST_CASE_METHOD(RemoteMessageTest,
    "catch2_IncompatibleKeys", "[RMsg]")
{
    // More arguments than keys
    std::tuple<std::int32_t, std::string, char> testTuple =
        std::make_tuple(123, "A string", 'x');
    {
        std::string format{
            "A format % with 2 % percents and 3 args" };
        auto rlm = CreateLogMessage(format, std::get<0>(testTuple),
            std::get<1>(testTuple), std::get<2>(testTuple));
        CHECK(std::string{ rlm.GetLogText() } == std::string{});
    }
    // More keys than arguments
    {
        std::string format{
            "A format % with 4 % percents and % 3 args %" };
        auto rlm = CreateLogMessage(format, std::get<0>(testTuple),
        std::get<1>(testTuple), std::get<2>(testTuple));
        CHECK(std::string{ rlm.GetLogText() } == std::string{});
    }
}
```

Registration with CTest

1. Include the CTest module
2. Add the manual tests
3. Fetch gtest
4. Add gtest tests
5. Fetch Catch2
6. Register Catch2 tests

```
include(CTest)
```

```
FetchContent_MakeAvailable(googletest)
include(GoogleTest)
gtest_discover_tests(${GTEST_TARGET})
```

```
include(Catch)
catch_discover_tests(${CATCH2_TARGET})
```

Running tests

```
alex@WLPF46XAH6:~/corecpp_ctest_talk/log_message_example/build/bin/Release$ ctest -N
Test project /home/alex/corecpp_ctest_talk/log_message_example/build/bin/Release
  Test #1: RemoteMessageTest.gtest_TextTruncation
  Test #2: RemoteMessageTest.gtest_VariadicArguments
  Test #3: catch2_NoKeyForArgs
  Test #4: catch2_IncompatibleKeys
  Test #5: NoArguments
  Test #6: EmptyMessage
```

```
alex@WLPF46XAH6:~/corecpp_ctest_talk/log_message_example/build/bin/Release$ ctest
Test project /home/alex/corecpp_ctest_talk/log_message_example/build/bin/Release
    Start 1: RemoteMessageTest.gtest_TextTruncation
1/6 Test #1: RemoteMessageTest.gtest_TextTruncation ......   Passed    0.00 sec
    Start 2: RemoteMessageTest.gtest_VariadicArguments
2/6 Test #2: RemoteMessageTest.gtest_VariadicArguments ...   Passed    0.00 sec
    Start 3: catch2_NoKeyForArgs
3/6 Test #3: catch2_NoKeyForArgs .........................   Passed    0.00 sec
    Start 4: catch2_IncompatibleKeys
4/6 Test #4: catch2_IncompatibleKeys .....................   Passed    0.00 sec
    Start 5: NoArguments
5/6 Test #5: NoArguments .................................   Passed    0.00 sec
    Start 6: EmptyMessage
6/6 Test #6: EmptyMessage ................................   Passed    0.00 sec

100% tests passed, 0 tests failed out of 6

Total Test time (real) =   0.01 sec
```

Build-and-test mode

- As the name implies – one step to build and test
- Enabled by adding 1 command to CMakeLists.txt
- Enforces to run the tests each time the project is built

```
Running test command: "/usr/bin/ctest" "--test-dir" "test"
Internal ctest changing into directory: /home/alex/corecpp_ctest_talk/log_message_example/build/test
Test project /home/alex/corecpp_ctest_talk/log_message_example/build/test
    Start 1: RemoteMessageTest.gtest_TextTruncation
1/6 Test #1: RemoteMessageTest.gtest_TextTruncation ......   Passed    0.00 sec
    Start 2: RemoteMessageTest.gtest_VariadicArguments
2/6 Test #2: RemoteMessageTest.gtest_VariadicArguments ...   Passed    0.00 sec
    Start 3: catch2_NoKeyForArgs
3/6 Test #3: catch2_NoKeyForArgs .........................   Passed    0.00 sec
    Start 4: catch2_IncompatibleKeys
4/6 Test #4: catch2_IncompatibleKeys .....................   Passed    0.00 sec
    Start 5: NoArguments
5/6 Test #5: NoArguments .................................   Passed    0.00 sec
    Start 6: EmptyMessage
6/6 Test #6: EmptyMessage ................................   Passed    0.00 sec

100% tests passed, 0 tests failed out of 6

Total Test time (real) =   0.01 sec
```

```
# For command line parsing
FetchContent_Declare(cxxopts GIT_REPOSITORY https://github.com/jarro2783/cxxopts.git GIT_TAG master)

enable_testing()

framework.cpp)
```

```
alex@WLPF46XAH6:~/corecpp_ctest_talk/log_message_example$ ctest --build-and-test . ./build \
> --build-generator Ninja --test-command ctest --test-dir test
```

More useful features

- Filter and shuffle tests
- Repeat tests
- Stop or rerun on failure
- Run tests in parallel
- Specify timeout

Test coverage

Challenge

- Hard to track what parts of code are covered by tests
- Tech debt evaluation (don’t shoot me)
- Requirements for code coverage in a regulated environment

Solution (Linux)

- LCOV - a graphical frontend for gcov
- During the tests run, coverage data is created
- Metrics are collected into a dedicated file
- A HTML report is generated
- Should be compiled in debug

Setting up the coverage report

Top CMakeLists.txt

```
list(APPEND CMAKE_MODULE_PATH "${CMAKE_SOURCE_DIR}/cmake")
```

test/CMakeLists.txt

```
include(Coverage)
RunCoverageReport(${PROJ_NAME})
```

Build command

```
alex@WBWIIL5Q2MDN2:~/corecpp_ctest_talk/lcov_example$ cmake -G Ninja \
> -S . -B build -DCMAKE_BUILD_TYPE=Debug && \
> cmake --build build -t coverage
```

cmake/Coverage.cmake

```
function(RunCoverageReport target)
    find_program(LCOV_PATH lcov REQUIRED)
    find_program(GENHTML_PATH genhtml REQUIRED)

    add_custom_target(coverage
        COMMENT "Running coverage for ${target}..."
        COMMAND ${LCOV_PATH} -d . --zerocounters
        COMMAND $<TARGET_FILE:${target}>
        COMMAND ${LCOV_PATH} -d . --capture -o coverage.info
        COMMAND ${LCOV_PATH} -r coverage.info '/usr/include/*'
            -o filtered.info
        COMMAND ${GENHTML_PATH} -o coverage filtered.info
            --legend
        COMMAND rm -rf coverage.info filtered.info
        WORKING_DIRECTORY ${CMAKE_BINARY_DIR}
        )
endfunction()
```

Coverage results

LCOV - code coverage report

Current view: top level
Test: filtered.info
Test Date: 2024-11-11 09:30:39
Legend: Rating: low: < 75 % | medium: >= 75 % | high: >= 90 %

	Coverage	Total	Hit
Lines:	82.4 %	131	108
Functions:	88.4 %	43	38

Directory	Line Coverage			Function Coverage		
	Rate	Total	Hit	Rate	Total	Hit
catch2/	0.0 %	2		0.0 %	1	
catch2/internal/	93.0 %	57	53	96.6 %	29	28
/home/alex/corecpp_ctest_talk/coverage_example/src/	66.0 %	50	33	70.0 %	10	7
/home/alex/corecpp_ctest_talk/coverage_example/test/	100.0 %	22	22	100.0 %	3	3

Generated by: LCOV version 2.2-beta

	Coverage	Total	Hit
Lines:	60.5 %	43	26
Functions:	57.1 %	7	4

```
        :
        :       template <class ...Args>
        3 :     RemoteLogMessage(const std::int32_t _messageId,
        :               const LogLevel _severity, const std::string& _format, Args&&... _args)
        3 :             : m_messageType{ _severity }
        3 :             , m_messageID{ _messageId }
        :       {
        3 :             m_header.extension.messageCode = MessageID::LOG_DATA;
        3 :             m_contentHeader.commandType = CommandType::N;
        3 :             m_contentHeader.numOfAttributes = 1;
        3 :             m_content.subject = MessageSubject::UNKNOWN_SUBJECT;
        3 :             m_content.attribute = 0;
        3 :             m_content.index = 0;
        3 :             m_content.dataType = DataType::BLOCK;
        :
        3 :             auto str = Format(_format, std::forward<Args>(_args)...);
        3 :             if (str.size() == 0)
        :               {
        3 :                     m_numberOfElementsToRead = 0;
        :               }
        :               else
        :               {
        0 :                     m_numberOfElementsToRead = (str.size() < MAX_LOG_MESSAGE_LENGTH - 1)
        0 :                             ? static_cast<std::int32_t>(str.size() + 1)
        :                               : MAX_LOG_MESSAGE_LENGTH;
        :               }
        :
        3 :             std::memcpy(m_logData, str.c_str(), m_numberOfElementsToRead);
        :
        3 :             SetDataLength(static_cast<std::uint32_t>(sizeof(m_messageType) +
        :                       sizeof(m_messageID) + sizeof(m_numberOfElementsToRead) +
        :                       sizeof(m_logData)));
        3 :     }
```

Summary

Purpose

- Automate testing in CMake-based projects

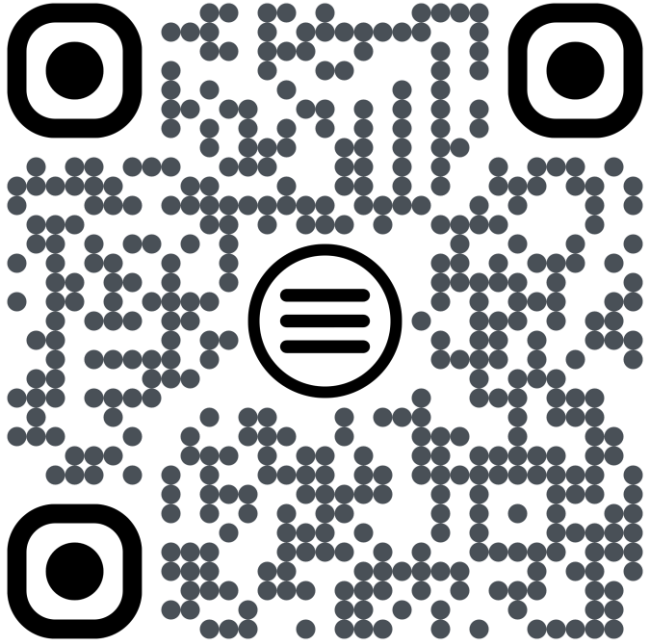
Key features

- Framework-agnostic
- Parallel test execution
- Generate detailed test reports

Benefits

- Simplifies testing process
- Standard way to manage and track test results
- Ensures code changes are safe

Thank you



My e-card

https://github.com/alexkushnir/corecpp_ctest_talk