



Cmake Build Framework – Part 2

C++



Algorithmic Trading & Quant Research Hub



C++ Set-Up for Algo Quant Trading
By Nicholas Burgess

C++ Set-Up for Algo Quant Trading



➤ Part 1 – Visual Studio for Windows

- Online C++ Emulators & Code Snippets
- Visual Studio Projects & Solutions
- C++ Building, Compilation & Linking

➤ Part 2 – CMake for Cross-platform Builds

- The CMake Build System
- How to use CMake
- Build Environments & Compilers

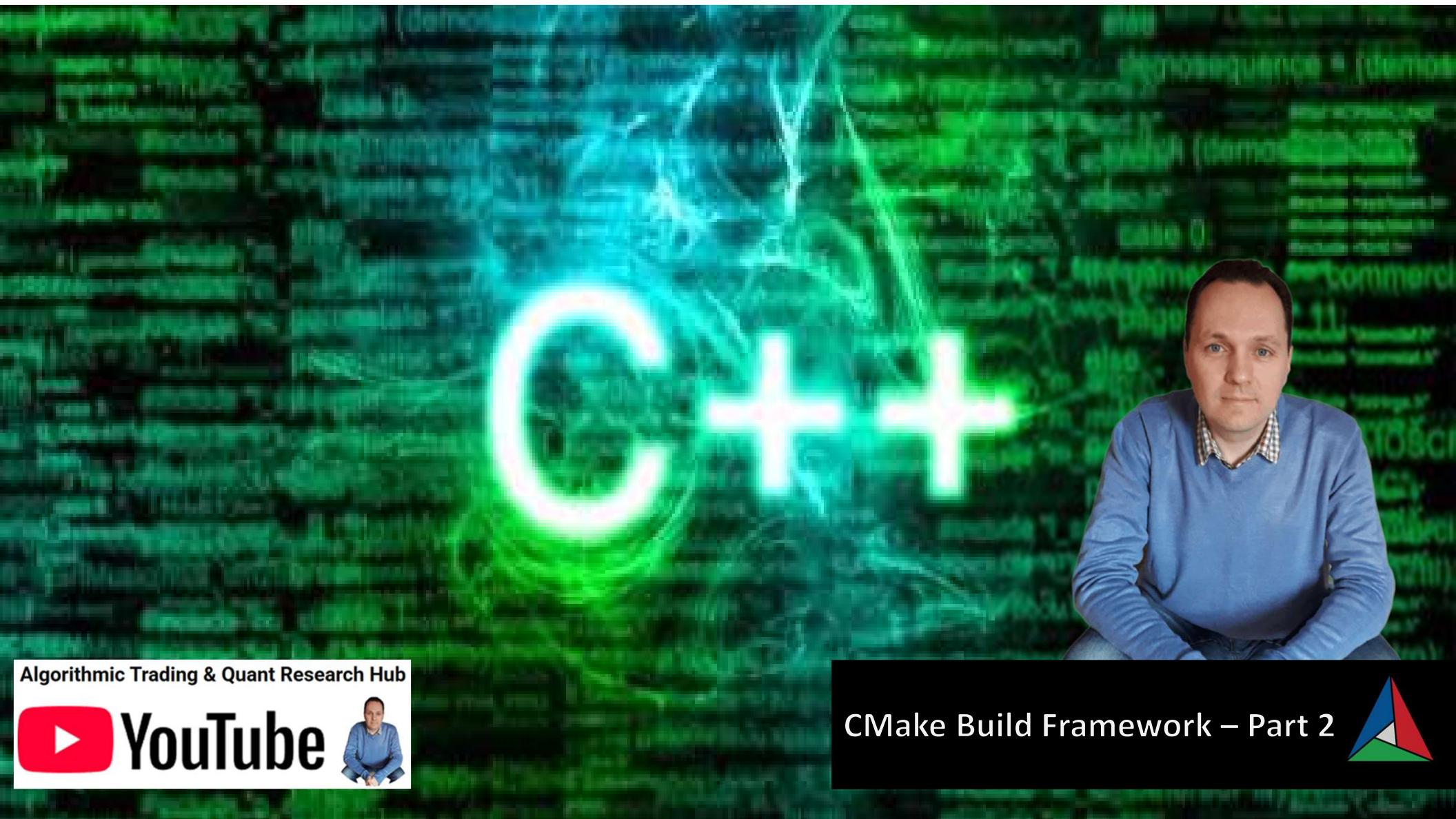


Example: Visual Studio & CMake

[https://github.com/nburgessx/QuantResearch
tree/main/CMake%20Examples](https://github.com/nburgessx/QuantResearch/tree/main/CMake%20Examples)

Algorithmic Trading & Quant Research Hub





CMake Build Framework – Part 2





CMake Cross-Platform Build System

CMake – What it is and what it does

- A **cross-platform** build system – not a compiler
- It uses platform-independent configuration files, **CMakeLists.txt**
- Generates native build files e.g. Visual Studio Solutions, Linux Make files, Ninja files, macOS Xcode projects
- Available as part of Visual Studio, see Tools -> Command line -> Developer Command Prompt

How to generate a solution File for Visual Studio?

- Create the necessary CMakeLists.txt files
- Open Visual Studio command line and type:

```
cmake -G "Visual Studio 17 2022" <path-to-project-root>
```



CMake Config Files – CMakeLists.txt

CMake Commands for CMakeLists.txt

➤ Solution Config File

- Name the solution file ([project](#))
- Specify what projects to include ([add_subdirectory](#))

➤ Project Config Files

- Name the project ([project](#))
- Specify the project type and list the headers and source files to include ([add_library](#) | [add_executable](#))
- Provide the path to the include folder(s) and header files ([target_include_directories](#))
- List any dependency projects to include ([target_link_libraries](#))



Example: Create Solution File

- Consider a simple C++ maths library where the main project is called **Analytics** that depends on two projects named **Addition** and **Subtraction**. The folder structure looks as follows,

MathLibrary (Root Folder)

CMakeLists.txt

Analytics

CMakeLists.txt | Main.cpp

Addition

CMakeLists.txt | Add.h | Add.cpp

Subtraction

CMakeLists.txt | Subtract.h | Subtract.cpp

- The solution root folder and each project folder requires a **CMakeLists.txt** config file
- The config file defines the **project type** and specifies the **include paths** and **project dependencies**

Solution Config File, CMakeLists.txt

```
1  cmake_minimum_required(VERSION 3.20)
2
3  project(MathLibrary LANGUAGES CXX)
4
5  # ---- Language standard ----
6  set(CMAKE_CXX_STANDARD 17)
7  set(CMAKE_CXX_STANDARD_REQUIRED ON)
8
9  # ---- Targets ----
10 add_subdirectory(Addition)
11 add_subdirectory(Subtraction)
12 add_subdirectory(Analytics)
```

- `project` – Name of the solution file
- `add_subdirectory` – List project folders to include

Main Project Config File, CMakeLists.txt

```
1 add_executable(Analytics
2           main.cpp
3 )
4
5 target_link_libraries(Analytics
6           PRIVATE
7           Addition
8           Subtraction
9 )
```

➤ [add_executable](#)

- Creates project that outputs an executable called Analytics.
- List all the .h and .cpp files to include.

➤ [target_link_libraries](#)

- List the project name then the dependency projects to include
- Here we add the addition and subtraction projects to the analytics project



Dependency Project Config File, CMakeLists.txt

```
1 add_library(Addition STATIC
2     add.h
3     add.cpp
4 )
5
6 target_include_directories(Addition
7     PUBLIC
8         ${BUILD_INTERFACE:${CMAKE_CURRENT_SOURCE_DIR}}
9 )
```

➤ add_library

- Creates a project named Addition. Use **STATIC** to generate a .lib and **SHARED** to generate a .dll
- List all the .h and .cpp files to include.

➤ target_include_directories

- List the include directories for the Addition project
- **\$(CMAKE_CURRENT_SOURCE_DIR)** means use the current folder



Generating the Visual Studio Solution File

How to generate the Visual Studio solution and project files?

- After creating the necessary CMakeLists.txt configuration files
- Open Visual Studio command prompt and navigate to the solution root folder
- Type **mkdir build** to create a folder called ‘build’
- Navigate to the build folder **cd build**

```
cmake -G "Visual Studio 17 2022" <path-to-project-root>
```

- To generate the solution file type: **cmake –G “Visual Studio 17 2022” ..**
- Note “..” means the root project is up one folder level

How to generate the native build projects for Linux, macOS and other platforms and compilers?

- Change the name of the compiler from “**Visual Studio 17 2022**” to the compiler of your choice
- Examples: For Linux “**Unix Makefiles**” or “**Ninja**” and for macOS use “**Xcode**”



CMake Resources





Getting Started with CMake

AlgoQuantHub Weekly Deep Dive



**Professional C++ with CMake for
Quants & Algo Trading**

Link: <https://algoquanthub.beehiiv.com/p/professional-c-with-cmake-for-quants-algo-trading>

Examples: <https://github.com/nburgessx/QuantResearch/tree/main/CMake%20Examples>



CMake Tutorial – cmake.org

The screenshot shows the CMake Tutorial page. At the top, there's a navigation bar with the CMake logo and "latest release (4.2.1)". Below it is a "Table of Contents" sidebar with sections like "CMake Tutorial" (Introduction, Steps), "Previous topic" (CPack WIX Generator), "Next topic" (Step 0: Before You Begin), "This Page" (Show Source, Quick search), and a "Go" button. The main content area has a title "CMake Tutorial" and a "Introduction" section. It describes the tutorial as a step-by-step guide for common build system issues. The "Steps" section lists two main steps: "Step 0: Before You Begin" and "Step 1: Getting Started with CMake", each with several sub-sections.

Table of Contents	<h2>CMake Tutorial</h2> <h3>Introduction</h3> <p>The CMake tutorial provides a step-by-step guide that covers common build system issues that CMake helps address. Seeing how various topics all work together in an example project can be very helpful.</p> <h3>Steps</h3> <p>The tutorial source code examples are available in this archive. Each step has its own subdirectory containing code that may be used as a starting point. The tutorial examples are progressive so that each step provides the complete solution for the previous step.</p> <ul style="list-style-type: none">• Step 0: Before You Begin<ul style="list-style-type: none">◦ Getting the Tutorial Exercises◦ Getting CMake◦ CMake Generators◦ Single and Multi-Configuration Generators◦ Other Usage Basics◦ Try It Out◦ Getting Help and Additional Resources• Step 1: Getting Started with CMake<ul style="list-style-type: none">◦ Background◦ Exercise 1 - Building an Executable◦ Exercise 2 - Building a Library◦ Exercise 3 - Linking Together Libraries and Executables◦ Exercise 4 - Subdirectories
-------------------	--

- [CMake Tutorial – cmake.org](https://cmake.org/cmake/help/book/mastering-cmake/cmake/Help/guide/tutorial/index.html)
- Provides a step-by-step guides and tutorials on how to use CMake
- Link: <https://cmake.org/cmake/help/book/mastering-cmake/cmake/Help/guide/tutorial/index.html>

C++

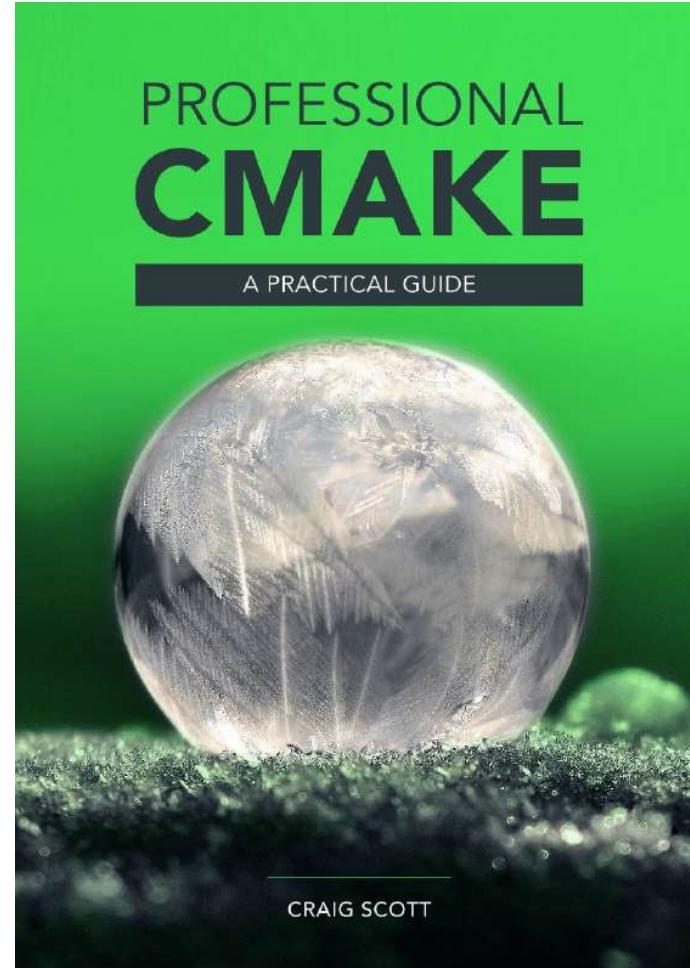
C++

C++

C++

Professional CMake

- Professional CMake – A Practical Guide
 - Free Book
 - By Craig Scott
 - <https://crascit.com/professional-cmake/>





More Info ...



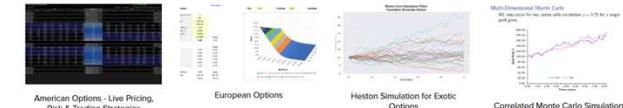
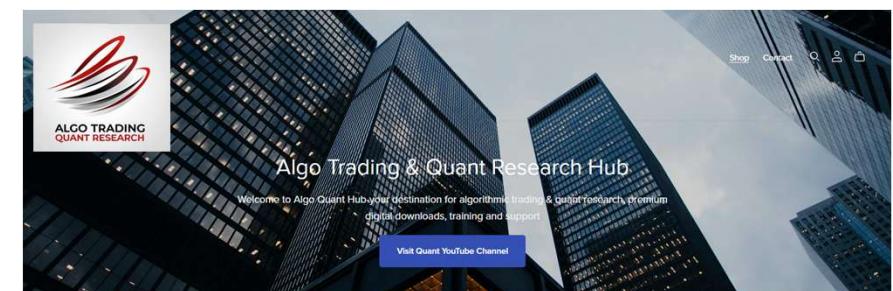


Subscribe to my Quant Newsletter
<https://algoquanthub.beehiiv.com/subscribe>

Subscribe to my Quant YouTube Channel
<https://www.youtube.com/@algoquanthub>

Algo Trading & Quant Store
<https://payhip.com/AlgoQuantHub>

Follow me on LinkedIn
<https://www.linkedin.com/in/nburgessx>



Have questions or want further info?

Contact

LinkedIn: www.linkedin.com/in/nburgessx