



## MS Visual Studio – Part 1



**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

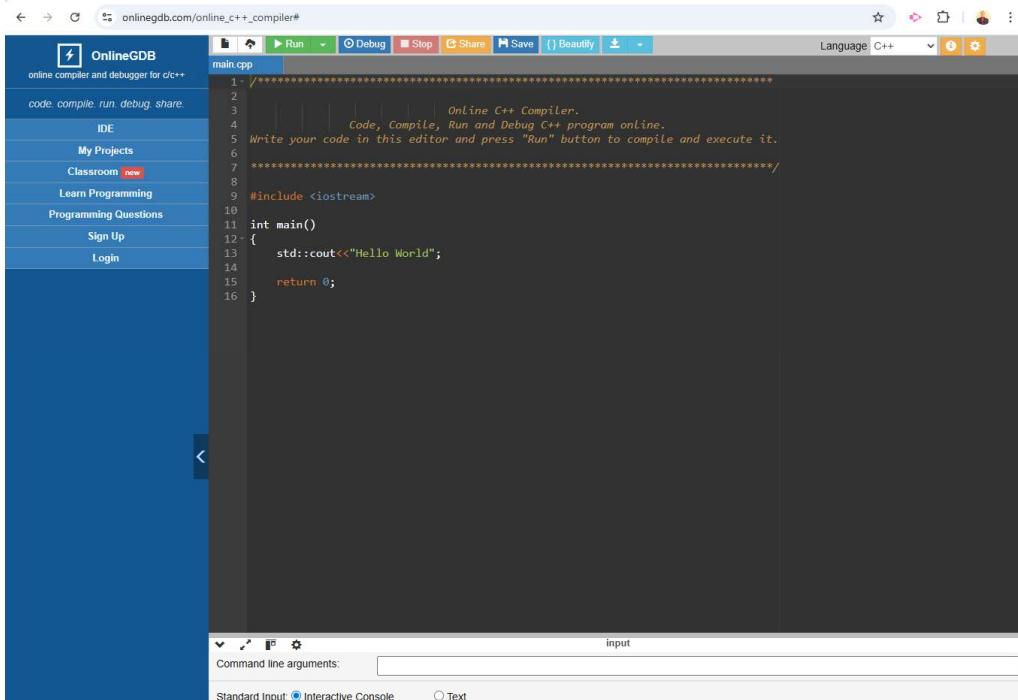
[https://github.com/nburgessx/QuantResearch  
tree/main/CMake%20Examples](https://github.com/nburgessx/QuantResearch/tree/main/CMake%20Examples)

Algorithmic Trading & Quant Research Hub





# C++ Online Emulator



- Can Select Language (top-right)  
**C++, Python, Java ...**
  - Can Select Version  
**C++14, C++17, C++23 ...**
  - Great for Learning C++ Syntax & Testing Ideas
  - Great for Code Snippets, Sharing & Debugging
  - Links nicely with GitHub



## Online C++ Compiler



# C++ Header and Source Files



## 1. Header Files (.h)

- Here we **declare** our functions, classes & interfaces

```
h  
int add(int a, int b);
```

## 2. Source Files (.cpp)

- **Define** and implement the header file declarations
- Contain the actual code logic

```
cpp  
int add(int a, int b) {  
    return a + b;  
}
```

## 3. Object Files (.obj)

- Source code from .cpp is compiled into an object file
- It is first translated into **assembly language** .asm
- The assembler then creates the object file .obj
- Object files contain **binary machine code** ready for linking

```
asm  
add:  
    mov eax, edi      ; move first argument a into register eax  
    add eax, esi      ; add second argument b to eax  
    ret               ; return value in eax
```

```
obj  
B8 ?? ?? ?? ?? ?? ; mov eax, ?  
01 F0              ; add eax, esi  
C3                 ; ret
```



# C++ Build Process – Compile & Link

## 1. Compile (cl.exe)

- Expands #include directives and checks syntax and types
- Converts each translation unit (.cpp file) into an object file

## 2. Link (link.exe)

- Verifies all symbols (i.e. functions and global variables) are defined
- Combines object files into a single file (**.exe | .lib | .dll**)

## 3. Generated Output Files

- |                                 |                                                                         |
|---------------------------------|-------------------------------------------------------------------------|
| ➤ Object Files:                 | These are compiled .cpp files ( <b>Windows .obj   Linux/macOS .o</b> )  |
| ➤ Static Library:               | A library ( <b>.lib</b> ) or collection of object files merged together |
| ➤ Dynamic Linked Library (DLL): | Compiled code loaded at runtime ( <b>.dll</b> )                         |
| ➤ Executable:                   | A fully linked program ready to run ( <b>.exe</b> )                     |



# Connecting Projects & Using Libraries



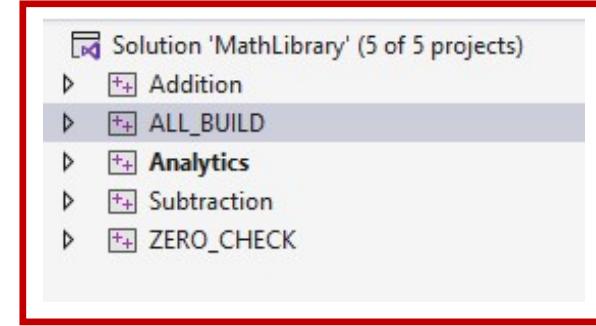
## Connecting Projects

- In C++ project folders are independent
- To share projects internally, we typically compile them as libraries (.lib)
- To use them we reference the **path to the include directory** (header files) and the **path to and name of the .lib file**
- To share projects externally we compile as them as a library, executable or DLL

## When using Libraries - Why are headers needed?

- **Headers** declare **what exists** (functions, classes, interfaces)
- **Libraries** contain **how it's implemented** (compiled machine code)

## Example: Solution & Projects



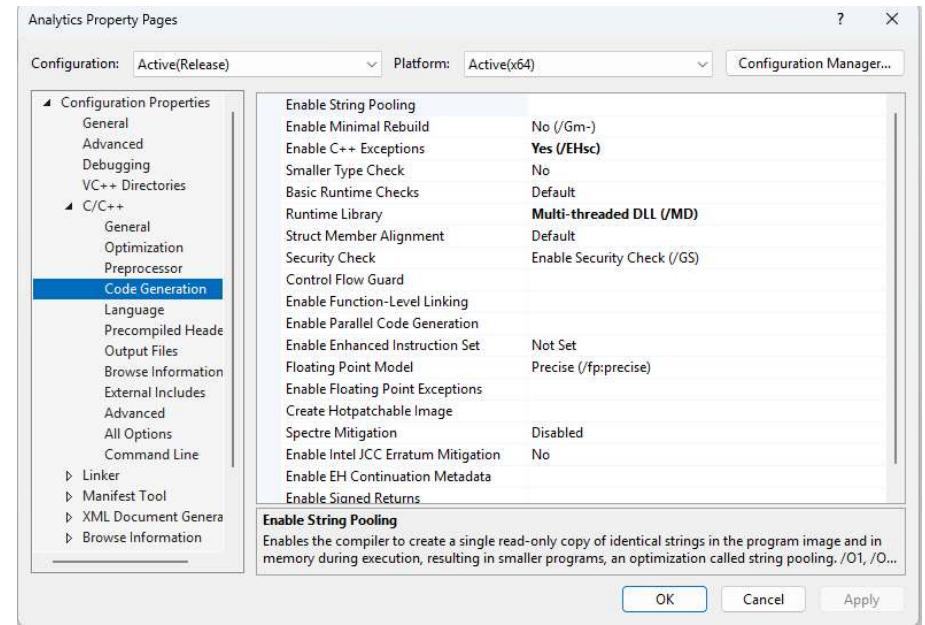
## Sharing Projects

- Header file path(s)
- Library file path



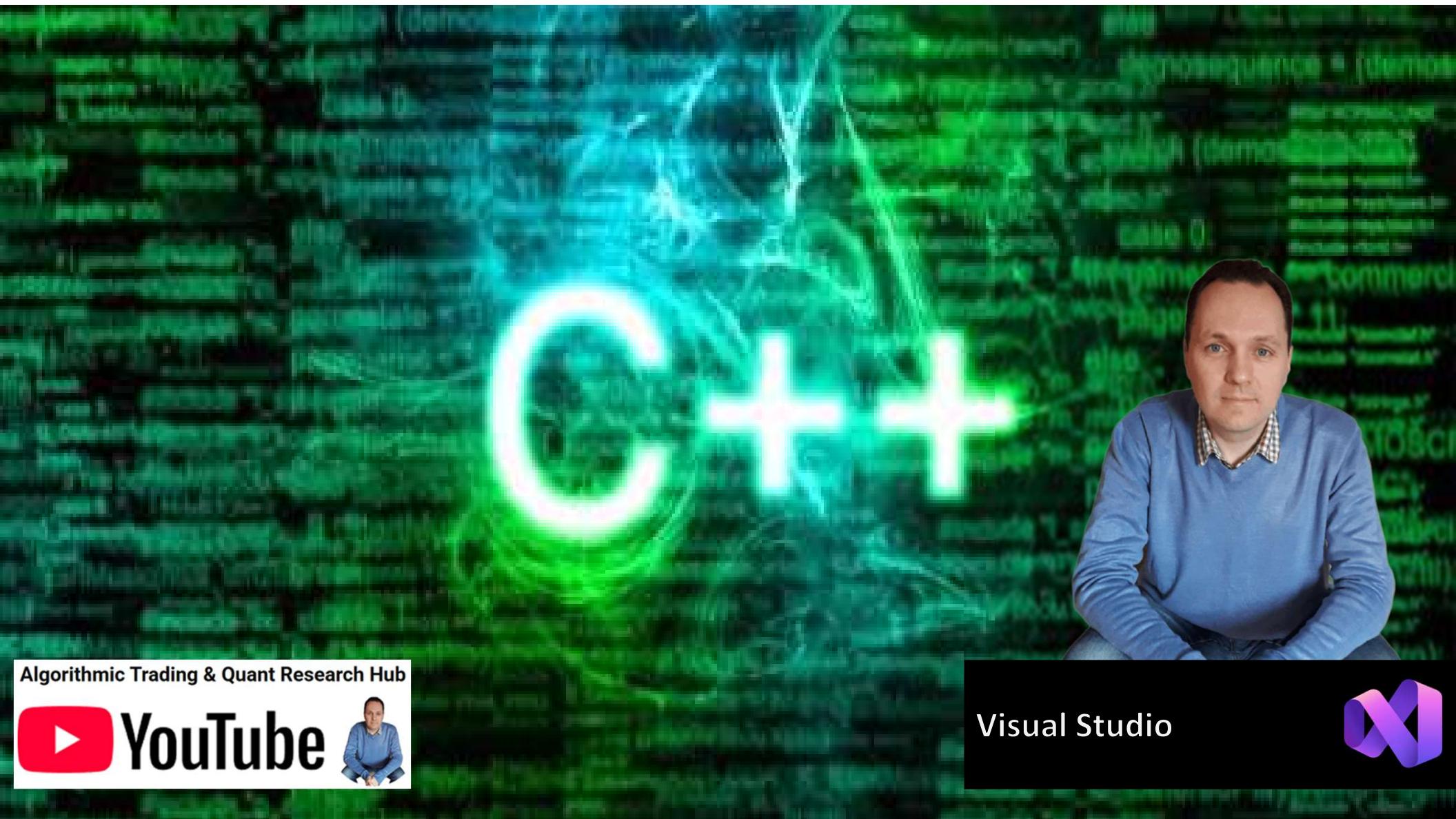
# Application Binary Interface (ABI)

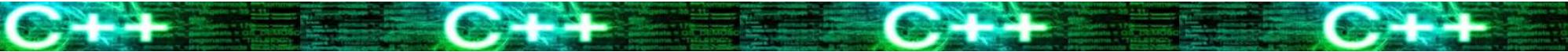
- ABI defines how project binaries are linked and how they manage memory
- Projects sharing runtime resources e.g. std::vector or FILE\* **must** use the same C++ Runtime library (CRT), which handles memory, I/O and startup support
- Dynamic Linkage (**/MD**) links against a shared C++ Runtime DLL (CRT)
- Static Linkage (**/MT**) embeds a private CRT into each binary
- Mixing /MD and /MT is unsafe – such code often builds successfully but fails and crashes at runtime



## Rule of thumb:

**/MD** → DLLs & large apps (shared runtime, one-heap)  
**/MT** → Fully self-contained tools (no shared ownership)





# Visual Studio

## ➤ Solution File

- Start-Up Project
- Project Dependencies (Build Order)
- Configuration
  - Debug, Release, Custom
  - Can Include/Exclude Projects

## ➤ Project Files

- Independent Code Project Groups

## ➤ Features

- Source Control – Git Integration
- Command Line – Dev Command Prompt
- External Tools – Custom Tools / Scripts
- Extensions – e.g. Incredibuild

The screenshot shows the Visual Studio interface with the following details:

- Solution Explorer:** Displays a solution named "MathLibrary" containing five projects: Addition, ALL\_BUILD, Analytics (selected), Subtraction, and ZERO\_CHECK.
- Output Window:** Shows the contents of main.cpp, which includes headers for iostream, add.h, and subtract.h. It defines a main function that adds and subtracts two double values (a=10, b=3) and prints the results to the console. It also includes a system("PAUSE") call.
- Toolbox:** Standard Visual Studio toolbox icons.
- Diagnostic Tools:** A sidebar on the right.
- Status Bar:** Shows the current line (Ln: 19), character (Ch: 14), and other status information.

A red box highlights the "Analytics" project in the Solution Explorer.

```
#include <iostream>
#include "add.h"
#include "subtract.h"

int main()
{
    double a = 10;
    double b = 3;

    std::cout << "Add: " << a << " + " << b
           << " = " << add(a, b) << std::endl;

    std::cout << "Subtract: " << a << " - " << b
           << " = " << subtract(a, b) << std::endl;

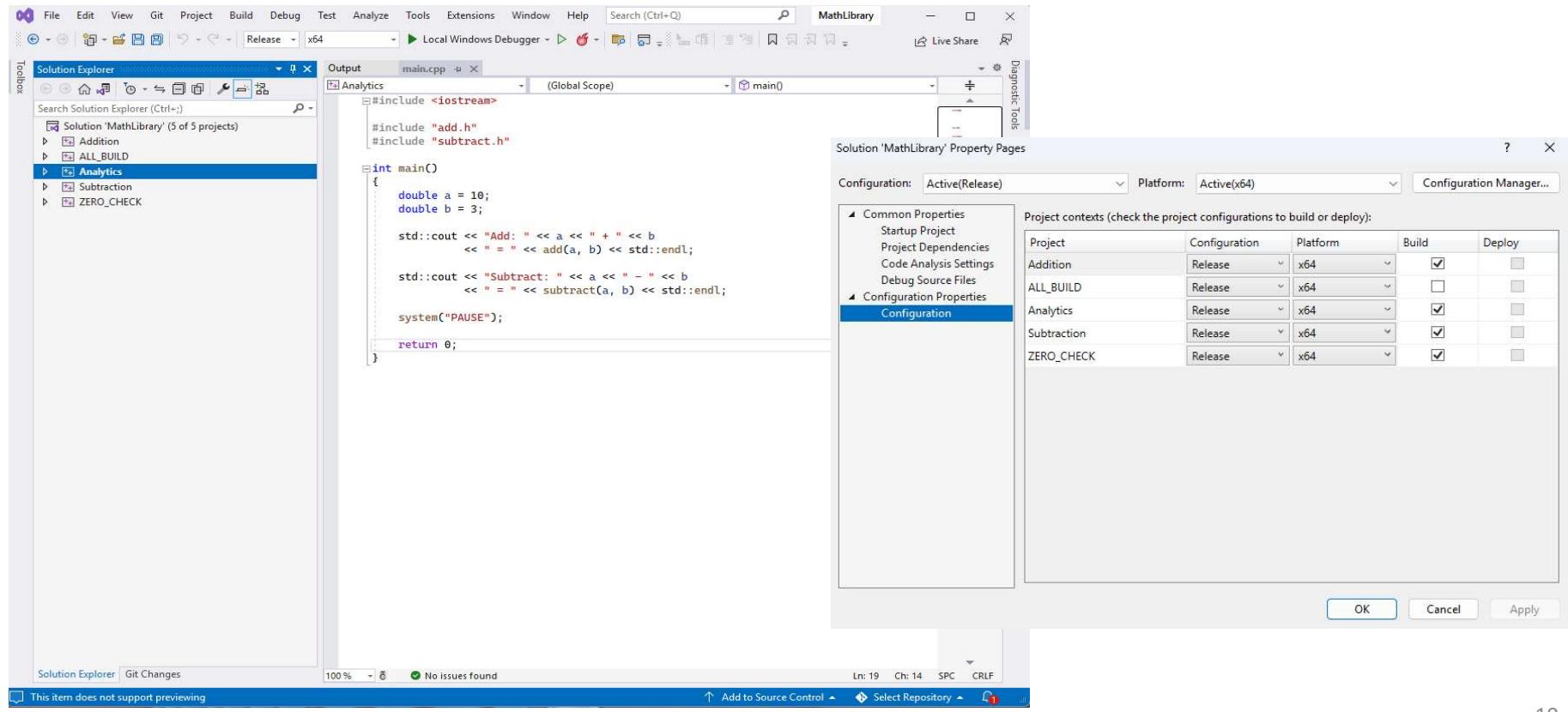
    system("PAUSE");
}

return 0;
}
```

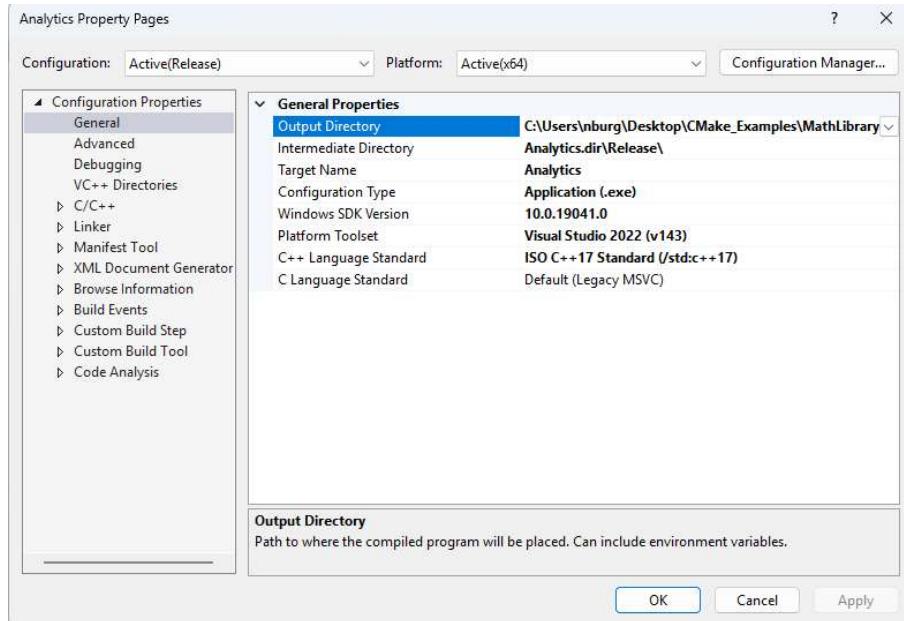


# Visual Studio Solution & Projects Files

These are XML files in disguise – Try opening them in notepad!



# Visual Studio Project Properties



## Output type

- Configuration Type (.lib | .exe | .dll)

## Where outputs go

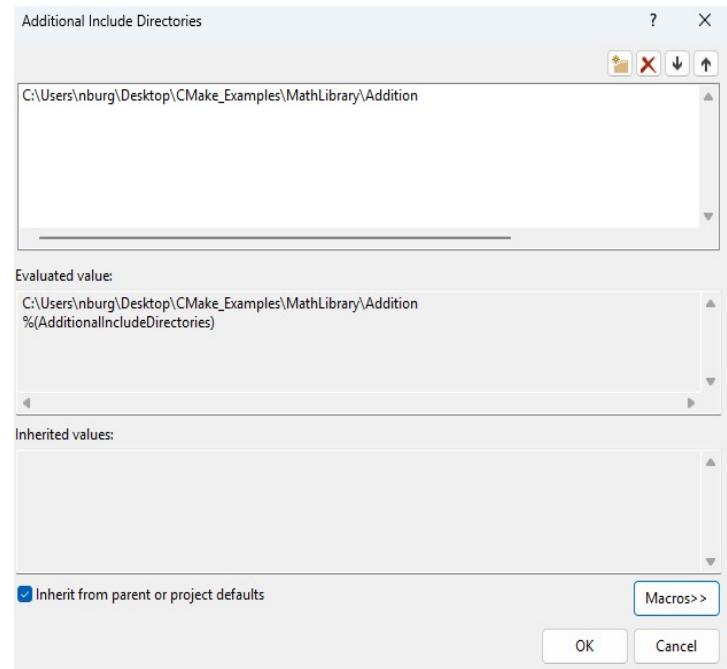
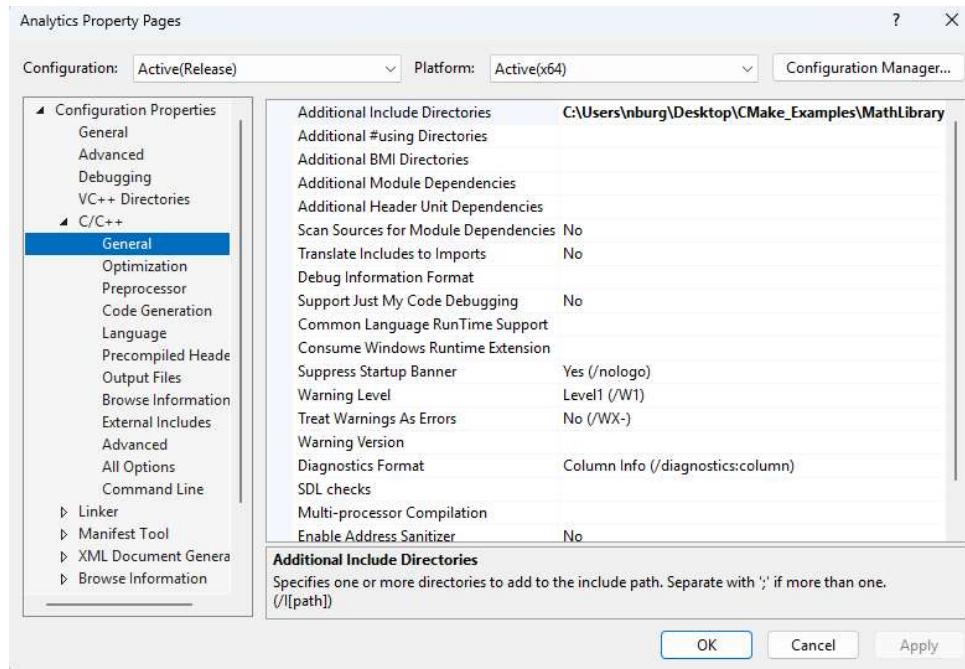
- Intermediate Directory (.obj)
- Output Directory (.lib | .exe | .dll)

## Solution and Project Files [TOP TIP]

- These are XML files that can be opened in Notepad
- XML supports extra features e.g. recursive file paths



# VS Project Properties – C/C++ Compiler

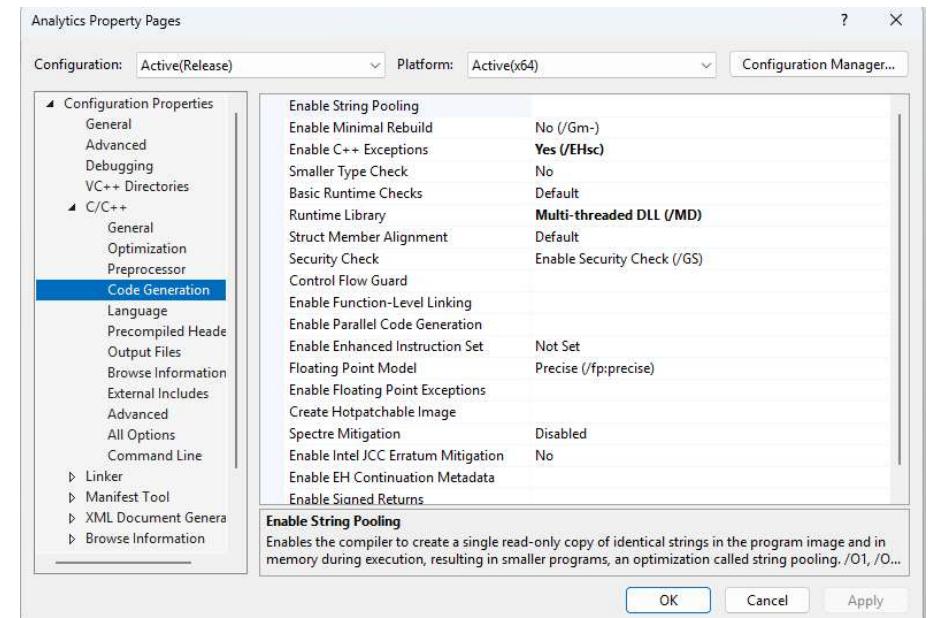
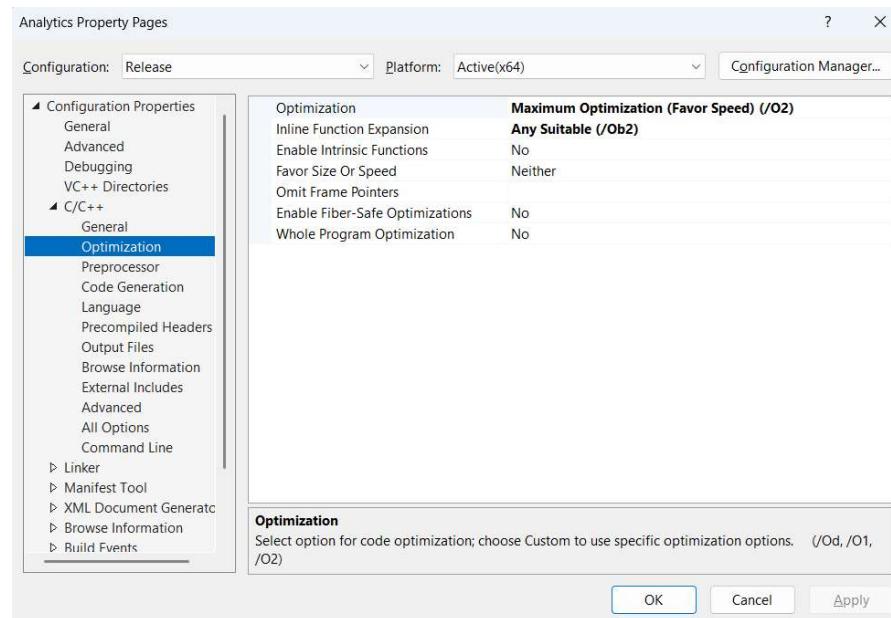


## File Path Macros [TOP TIP]

- Click the down arrow on any directory folder, then in the window pop-up press the “**Macros**” button
- View existing file path variables (macros) and/or add new ones e.g. \$(SolutionDir), \$(ProjectDir), ...

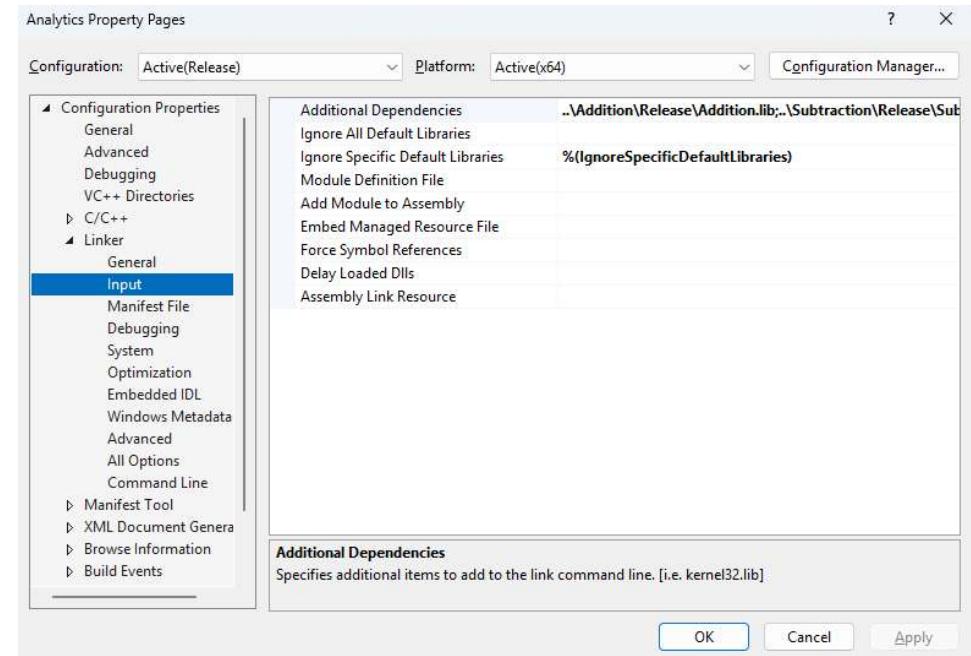
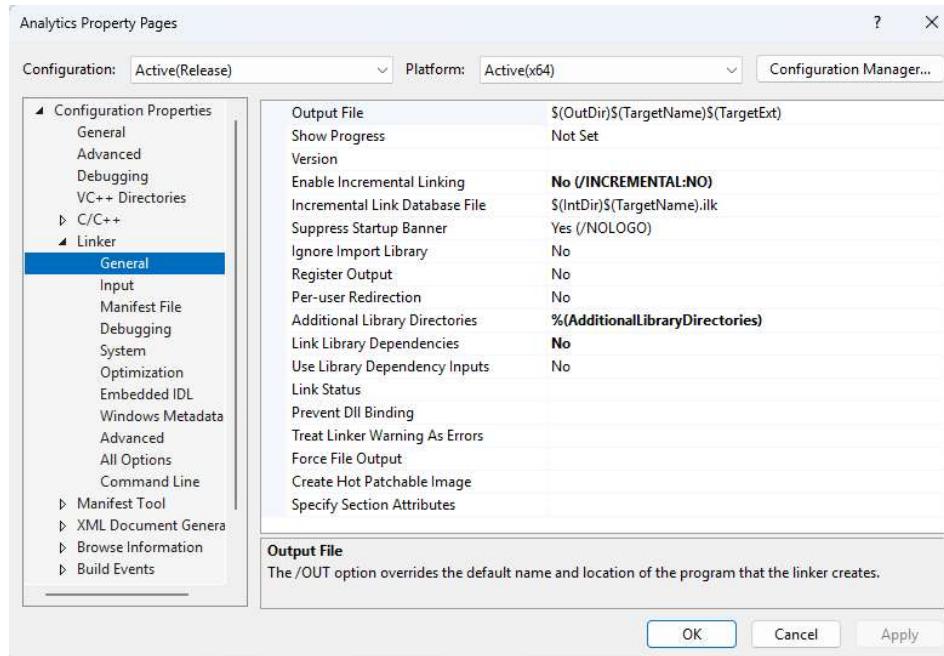


# VS Project Properties – C/C++ Compiler





# VS Project Properties – Linker/Librarian





# Summary – Key Project Properties



## ➤ General

- **Output Directory** – Specify output path
- **Configuration Type** – Specify the output file type .lib, .exe or .dll
- **C++ Language Standard** – C++14, C++17, C++20 ...

## ➤ C/C++ → General

- **Additional Include Directories** – To link projects, add include folder(s) here
- **Debug Information Format** – Edit and Continue (/ZI) this allows us to make minor modifications without rebuilding the project
- **Multi-processor Compilation (Yes /MP)** – allows parallel building of .cpp files

## ➤ C/C++ → Code Generation

- **Enable C++ Exceptions** – /Ehsc allows structured exception handling and helps prevent crashes
- **Runtime Library** – Here we must specify dynamic or static linking of CRT (/MD or /MT), defaults to /MD

## ➤ Linker → General:

- **Additional Library Directories** - To link projects, add path to .lib files here

## ➤ Linker → Input:

- **Additional Dependencies** – To link projects, specify .lib path here

## ➤ Linker → Debugging

- **Generate Debug Info** – To test and debug a release project select /DEBUG



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>



## Algorithmic Trading & Quant Research Hub

@AlgoQuantHub ·  
Algorithmic Trading & Quant Research .



Have questions or want further info?

## Contact

LinkedIn: [www.linkedin.com/in/nburgessx](https://www.linkedin.com/in/nburgessx)