Goals

1. While we ALL *love* Codio, it will be important to be able to develop, compile, and test code on your laptop, natively. Thus, we want to help you install a working C++ compiler, editor, and other tools on your laptop and have you write and run a simple "Hello world" application.

2. We would also like to review inheritance and polymorphism by having you run through the Lab 11 assignment on Codio. Let's do that first!
Part 1 - Review of Polymorphism
Can someone explain:

- Protected class members
- The order in which constructors of a Base, Child, and Grandchild class run
- What difference does the "virtual" keyword make in a function prototype of a base class member function?
- What is a pure virtual function?
- What is an abstract class?
Your Task

- Complete the two exercises in Codio Lab 11 (one analysis, one coding)
  - Since everyone may have different comfort levels, we'll let you work at your own pace and you can ask questions along the way. If we recognize common questions, we may pull the group together to discuss it as a large group
  - You need to complete this assignment to get credit for the lab.

- Then you can move on to installing your own toolchain (described on the next slides) which you can work through on your own. If at all possible, we'd like you to show us you can compile and run the "Hello World" application on your laptop, but realize that not everyone may finish (which is fine).
Part 2 - Tutorial of MS VS Code
MS Visual Studio Code

- Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.
- You will learn how to install and debug locally on your laptop in this lab.
Instructions Based on OS

- Windows (next slide - 9)
- Mac (slide 14)

If you already have your own toolchain installed, show us that you can compile and run a simple C++ program.
Using GCC with MinGW on Windows
(MacOS go to slide 14)

• In this tutorial, you’ll configure Visual Studio Code to use the GCC C++ compiler (g++) and GDB debugger from mingw-w64 to create programs that run on Windows.

• After configuring VS Code, you will compile and debug a simple Hello World program in VS Code.

• If you have any problems, feel free to check the VS Code documentation repository.
Prerequisites

• Install Visual Studio Code
• Install the C/C++ extension for VS Code. You can install the C/C++ extension by searching for 'c++' in the Extensions view.

![C/C++ extension in VS Code](image)

• Get the latest version of Mingw-w64 via MSYS2, which provides up-to-date native builds of GCC, Mingw-w64, and other helpful C++ tools and libraries. You can download the latest installer from the MSYS2 page or use this link to the installer.
• Follow the Installation instructions on the MSYS2 website to install Mingw-w64. Take care to run each required Start menu and pacman command.
Prerequisites

• Install the Mingw-w64 toolchain (pacman -S --needed base-devel mingw-w64-x86_64-toolchain). Run the pacman command in a MSYS2 terminal. Accept the default to install all the members in the toolchain group.

• Add the path to your Mingw-w64 bin folder to the Windows PATH environment variable by using the following steps:
  – In the Windows search bar, type 'settings' to open your Windows Settings.
  – Search for Edit environment variables for your account.
  – Choose the Path variable in your User variables and then select Edit.
  – Select New and add the Mingw-w64 destination folder path to the system path. The exact path depends on which version of Mingw-w64 you have installed and where you installed it. If you used the settings above to install Mingw-w64, then add this to the path: C:\msys64\mingw64\bin.
  – Select OK to save the updated PATH. You will need to reopen any console windows for the new PATH location to be available.
Check your MinGW

• To check that your Mingw-w64 tools are correctly installed and available, open a new Command Prompt (do NOT use the previous one you had open) and type:
  – gcc --version
  – g++ --version
  – gdb --version

• If you don't see the expected output or `g++` or `gdb` is not a recognized command, make sure your PATH entry matches the Mingw-w64 binary location where the compilers are located. If the compilers do not exist at that PATH entry, make sure you followed the instructions on the MSYS2 website to install Mingw-w64.

• If `gcc` has the correct output but not `gdb`, then you need to install the packages you are missing from the Mingw-w64 toolset.
  – Missing the `mingw-w64-gdb` package is one cause of the "The value of miDebuggerPath is invalid." message upon attempted compilation if your PATH is correct.
Create Hello World

• Follow Instructions on this link
Using GCC with MinGW on MacOS

• In this tutorial, you configure Visual Studio Code on macOS to use the Clang/LLVM compiler and debugger.

• After configuring VS Code, you will compile and debug a simple C++ program in VS Code.

• If you have any problems, feel free to check the VS Code documentation repository.
Prerequisites

- Install Visual Studio Code on macOS.
- Install the C++ extension for VS Code. You can install the C/C++ extension by searching for 'c++' in the Extensions view.

Ensure Clang is installed

- Clang may already be installed on your Mac. To verify that it is, open a macOS Terminal window and enter the following command:
  - `clang --version`
- If Clang isn't installed, enter the following command to install the command line developer tools:
  - `xcode-select --install`
Create Hello World

- Follow Instructions on [this link](#)