

```
// Michigan Robotics 102  
// Introduction to AI and Programming
```

```
// Making our first program in C++:  
// code, compile, run, repeat
```

```
#include <iostream>  
int main()  
{
```

Hello World!

```
    std::cout << "  
    ";
```

```
}
```

A simple program in C++

```
#include <iostream>
int main()
{
    std::cout << "Hello World! " ;
}
```

A simple program in C++

What is C++?

```
#include <iostream>
int main()
{
    std::cout << "Hello World! ";
}
```

C++

From Wikipedia, the free encyclopedia

"CXX" redirects here. For other uses, see [CXX](#) (disambiguation).

What is C++?

C++ (/sɪ plʌs plus/) is a general-purpose programming language created by [Bjarne Stroustrup](#) as an extension of the C programming language, or "C with Classes". The language has expanded significantly over time, and modern C++ now has object-oriented, generic, and functional features in addition to facilities for low-level memory manipulation. It is almost always implemented as a compiled language, and many vendors provide C++ compilers, including the Free Software Foundation, LLVM, Microsoft, Intel, Oracle, and IBM, so it is available on many platforms.^[1]

C++ was designed with an orientation toward system programming and embedded, resource-constrained software and large systems, with performance, efficiency, and flexibility of use as its design highlights.^[10] C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications,^[10] including desktop applications, video games, servers (e.g. e-commerce, web search, or databases), and performance-critical applications (e.g. telephone switches or space probes).^[11]

C++ is standardized by the International Organization for Standardization (ISO), with the latest standard version ratified and published by ISO in December 2020 as [ISO/IEC 14882:2020](#) (informally known as C++20).^[12] The C++ programming language was initially standardized in 1998 as [ISO/IEC 14882:1998](#), which was then amended by the C++03, C++11, C++14, and C++17 standards. The current C++20 standard supersedes these with new features and an enlarged standard library. Before the initial standardization in 1998, C++ was developed by Danish computer scientist Bjarne Stroustrup at Bell Labs since 1979 as an extension of the C language; he wanted an efficient and flexible language similar to C that also provided high-level features for program organization.^[13] Since 2012, C++ has been on a three-year release schedule^[14] with C++23 as the next planned standard.^[15]

Contents [hide]

1 History

- 1.1 Etymology
- 1.2 Philosophy

C++



The C++ logo endorsed by Standard C++

Paradigms	Multi-paradigm: procedural, functional, object-oriented, generic, modular
Family	C
Designed by	Bjarne Stroustrup
Developer	ISO/IEC JTC1 (Joint Technical Committee 1) / SC22 (Subcommittee 22) / WG21 (Working Group 21)
First appeared	1983; 38 years ago
Stable release	C++20 (ISO/IEC 14882:2020) / 15 December 2020; 8 months ago
Preview release	C++23 () / 18 June 2021; 2 months ago

What is C++?

High-level programming language



What is C++?



High-level programming language

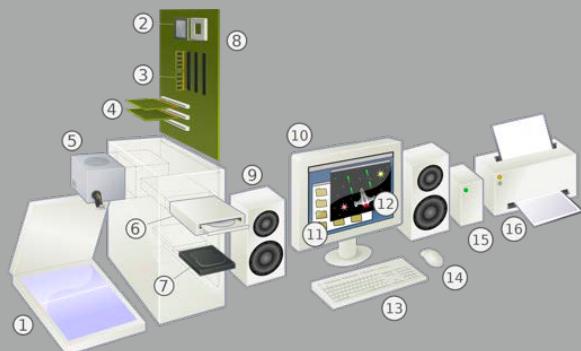


Computing Hardware

What is C++?



High-level programming language

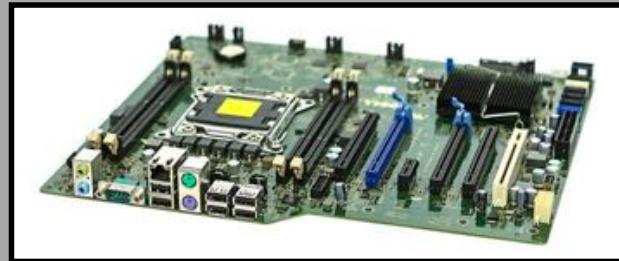


Computing Hardware

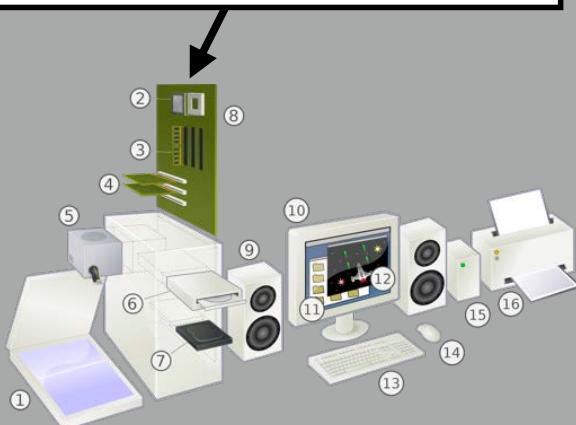
What is C++?



High-level programming language



Main circuit board
or “*Motherboard*”

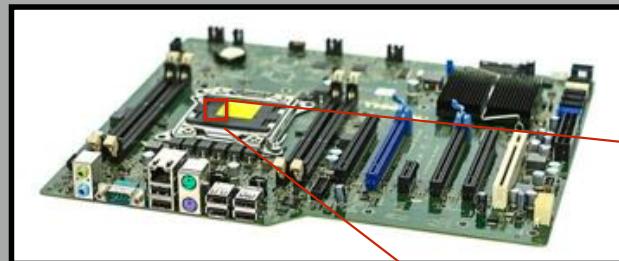


Computing Hardware

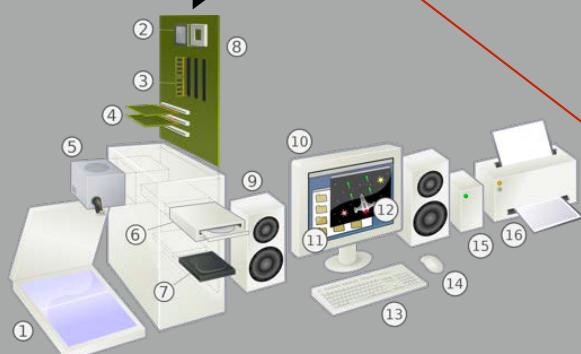
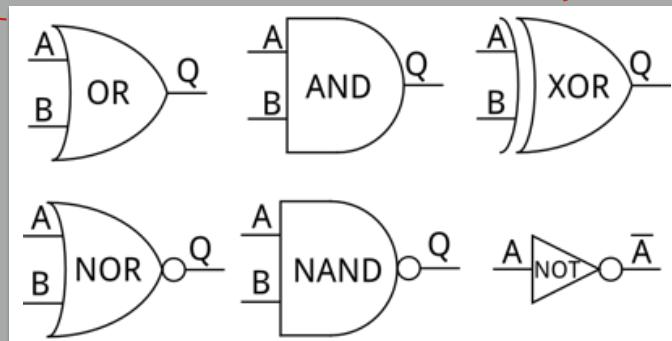
What is C++?



High-level programming language



Electronics that can carry out digital logic



Computing Hardware

What is C++?



High-level programming language

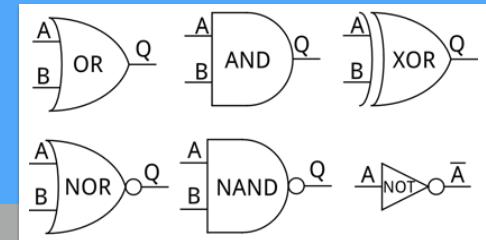
```
11101110  
01100100  
11001001  
11101110  
01100100
```

A program in binary code

Executable Software



Computing Hardware



Electronics that can carry out digital logic

What is C++?



High-level programming language

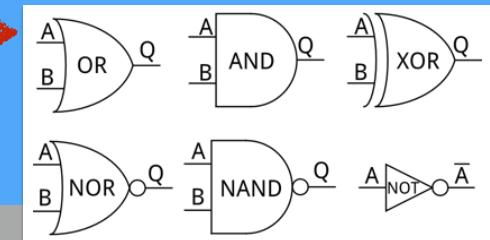
11101110
01100100
11001001
11101110
01100100

A program in binary code specifying a sequence of digital logic instructions that run on a computer

Executable Software



Computing Hardware



Electronics that can carry out digital logic

What is C++?

High-level programming language

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

Source Code

11101110
01100100
11001001
11101110
01100100

Executable Software



Computing Hardware

A program in readable text that can be compiled into a software executable

A program in binary code specifying a sequence of digital logic instructions that run on a computer

Electronics that can carry out digital logic

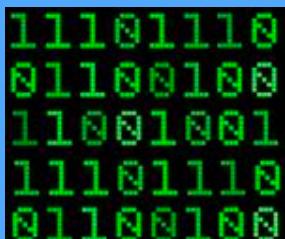
What is C++?

High “Coding” is writing source code

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

Source Code

A program in readable text that can be compiled into a software executable



11101110
01100100
11001001
11101110
01100100

Executable Software

A program in binary code specifying a sequence of digital logic instructions that run on a computer



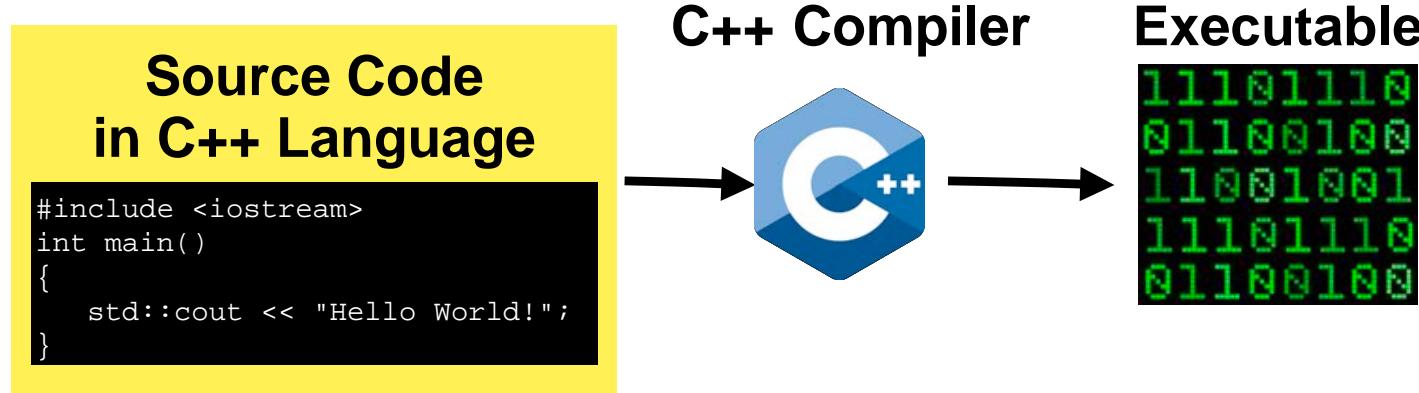
Computing Hardware

Electronics that can carry out digital logic

What is C++?

High-level programming language

Compiles source files into executable apps



What is C++?

High-level programming language

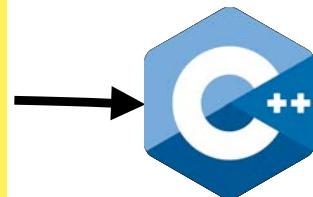
Portable across nearly all computers



Source Code
in C++ Language

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

C++ Compiler

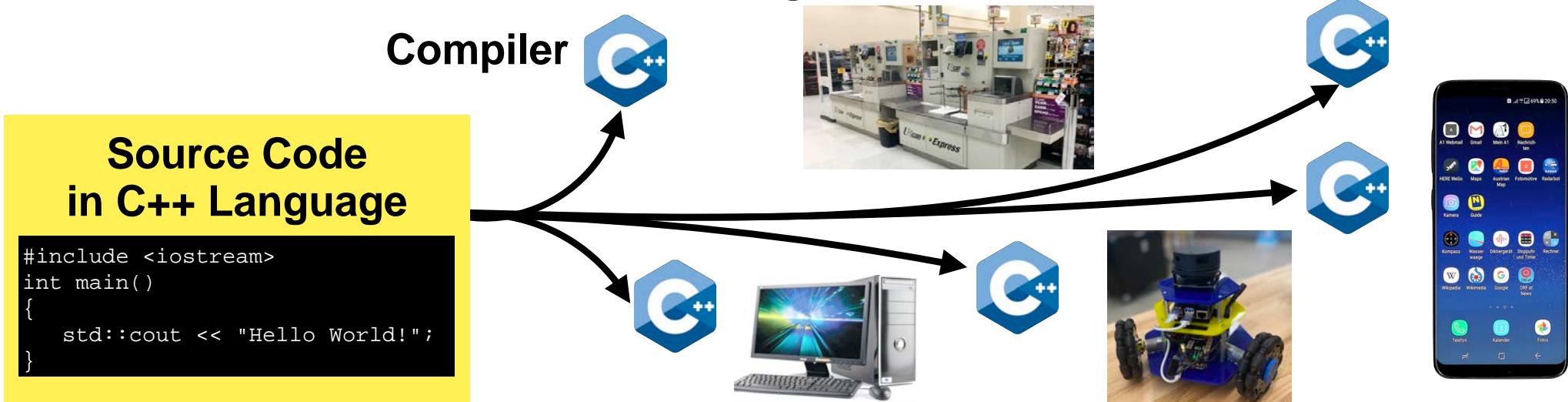


Michigan Robotics 102 - robotics102.org

What is C++?

High-level programming language

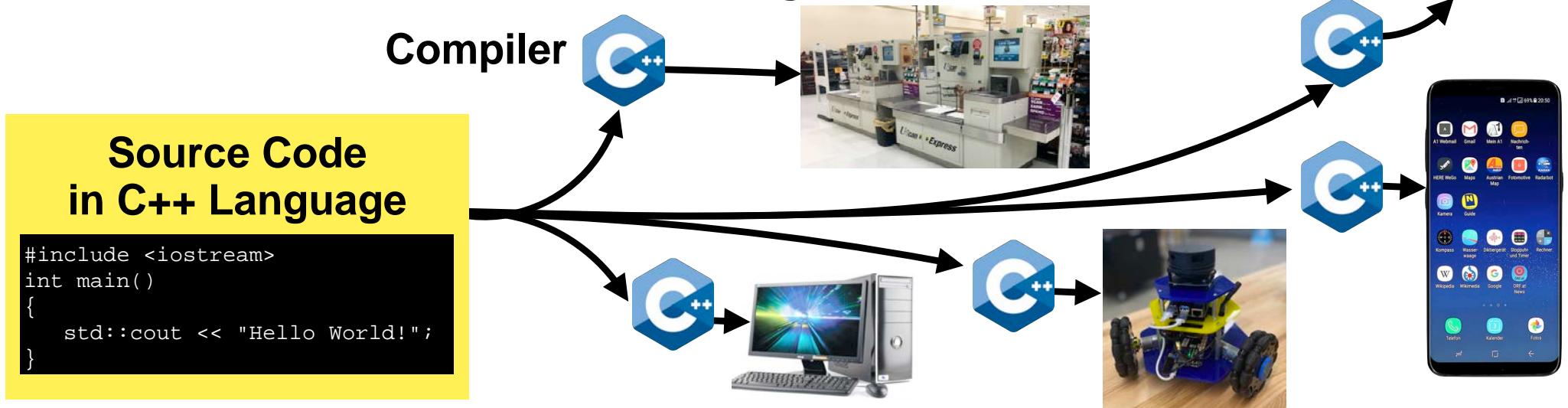
Portable across nearly all computers



What is C++?

High-level programming language

Portable across nearly all computers



What is C++?

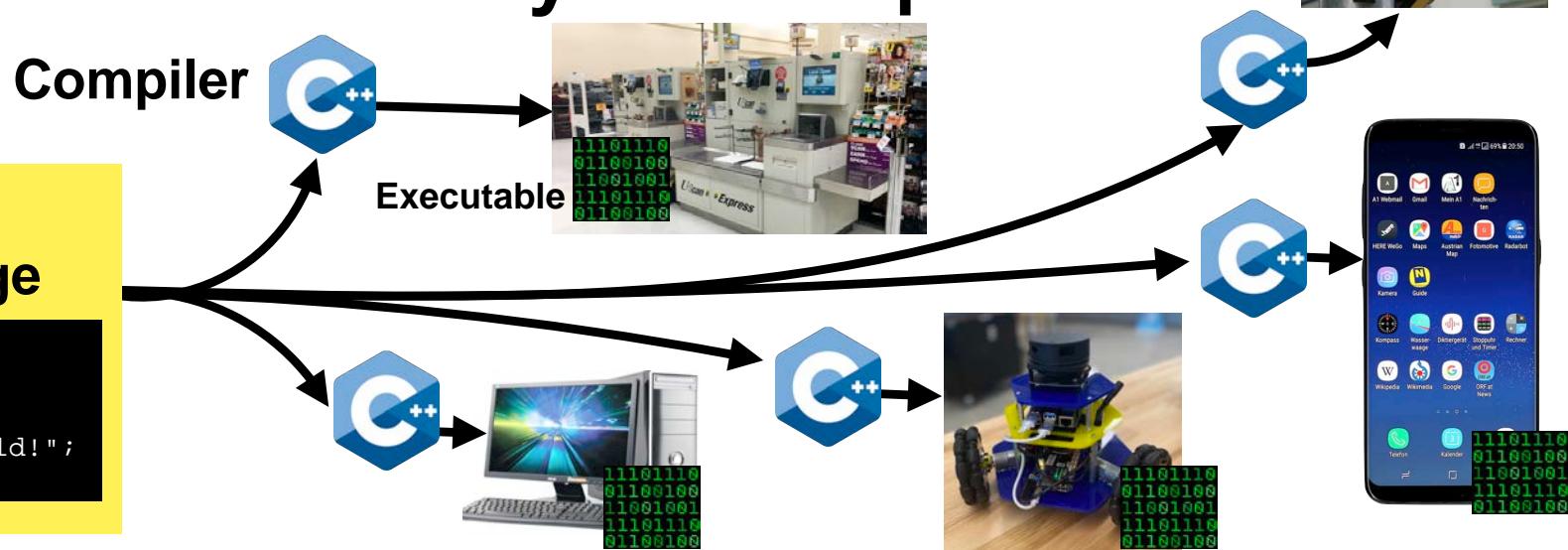


High-level programming language

Portable across nearly all computers

Source Code
in C++ Language

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```



Software Portability



It runs Doom!

Michigan Robotics 102 - robotics102.org



it runs doom



All Images Videos News Shopping More Tools Collections Satisfia



It Runs Doom | Know Your Meme
knowyourmeme.com



Programmer Plays Doom on a Pregnancy ...
PopularMechanics.com



DOOM will run on literally anything ...
gamercentral.com



run the game 'Doom' ...
ignitionmag.com



It Runs Doom!
whatshouldisurlife.com



The Treadmill at my school runs D...
reddit.com



Nintendo's Game & Watch can run Doom...
theverge.com



Chip From a \$15 Ikea Smart Lamp
IA.pcmag.com



It Runs Doom | Know Your Meme
knowyourmeme.com



Devices That Can Somehow Run Doom ...
vice.com



It Runs Doom | Rip It Apart - Jason's ...
ripitapart.com



It Runs Doom!
whatshouldisurlife.com



DOOM on a Digital Camera from 1998 ...
youtube.com



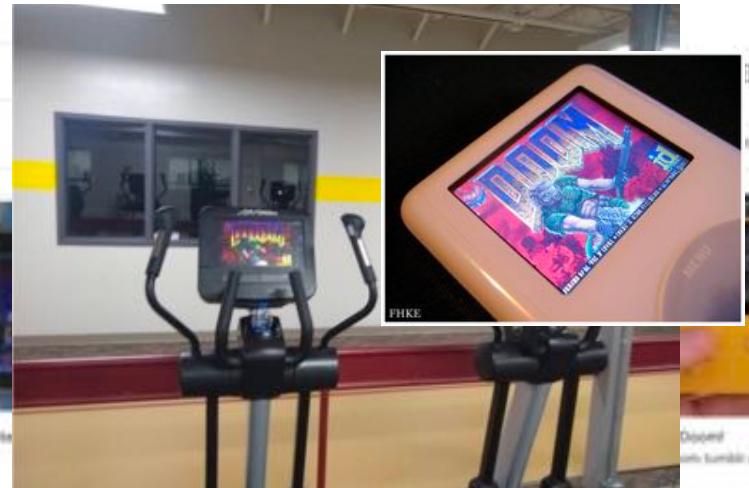
Yes, DOOM Can Run On A Pregnancy Test...
pcformatmagazine.com



If it has a processor,
reddit.com



it runs doom



Michigan Robotics 102 - robotics102.org

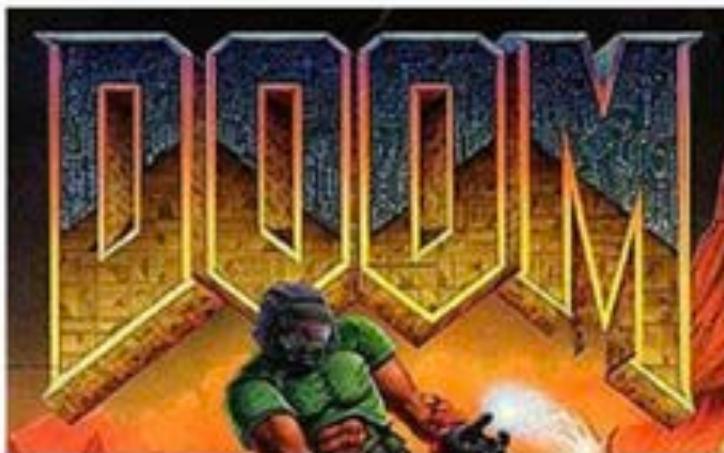
Neil's Place / DoomPhone

Technical Review: How I hacked an office telephone to play DOOM

Published 2017-08-01

Contents

1. Background
2. Hacking the phone
3. Make it run DOOM
 - Display Driver
 - Keypad Driver
 - Compiling DOOM
4. Conclusion



Background

In late 2017 I was a new hire on a small (maybe 10 person) IT team. We generally had an option to take hardware for ourselves if



What is C++?

High-level programming language



Portable across nearly all computers

Compiles source files into executable apps

What is C++?

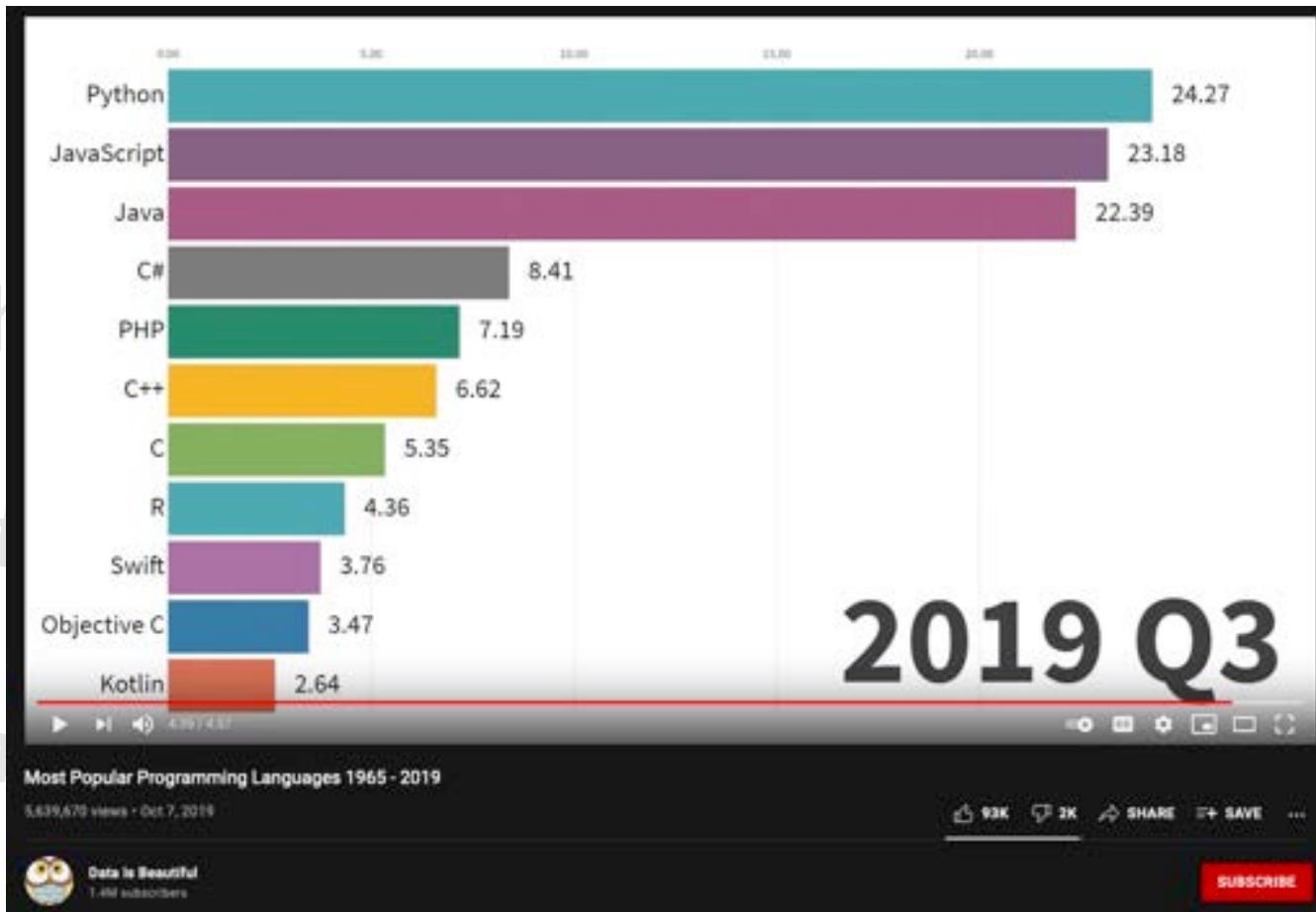
High-level programming language



Portable across nearly all computers

Compiles source files into executable apps

One of the most popular languages



One of the most popular languages



One of the most popular languages

A simple program in C++

How to code in C++ ?

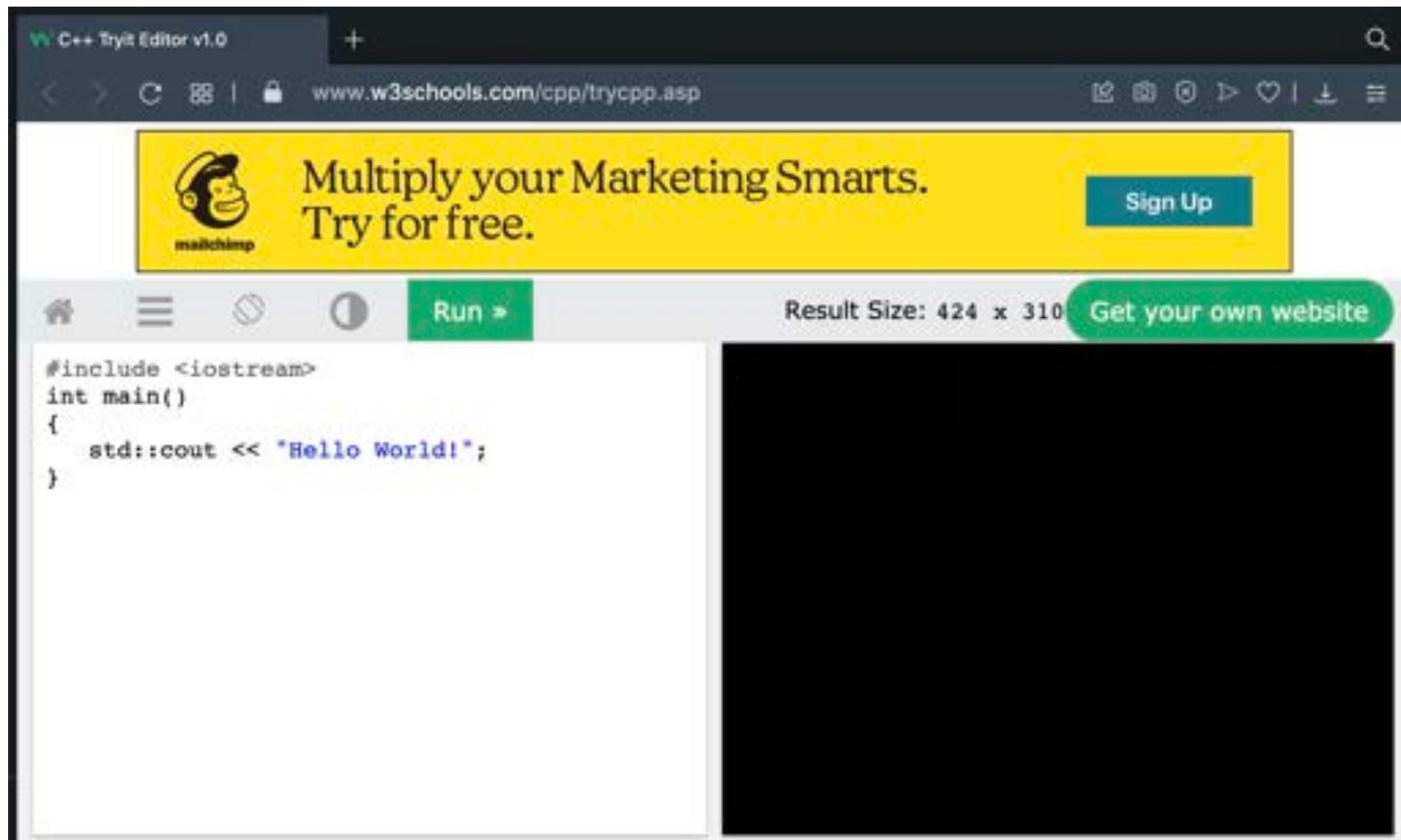
```
#include <iostream>
int main()
{
    std::cout << "Hello World! ";
}
```

How to code in C++ ?

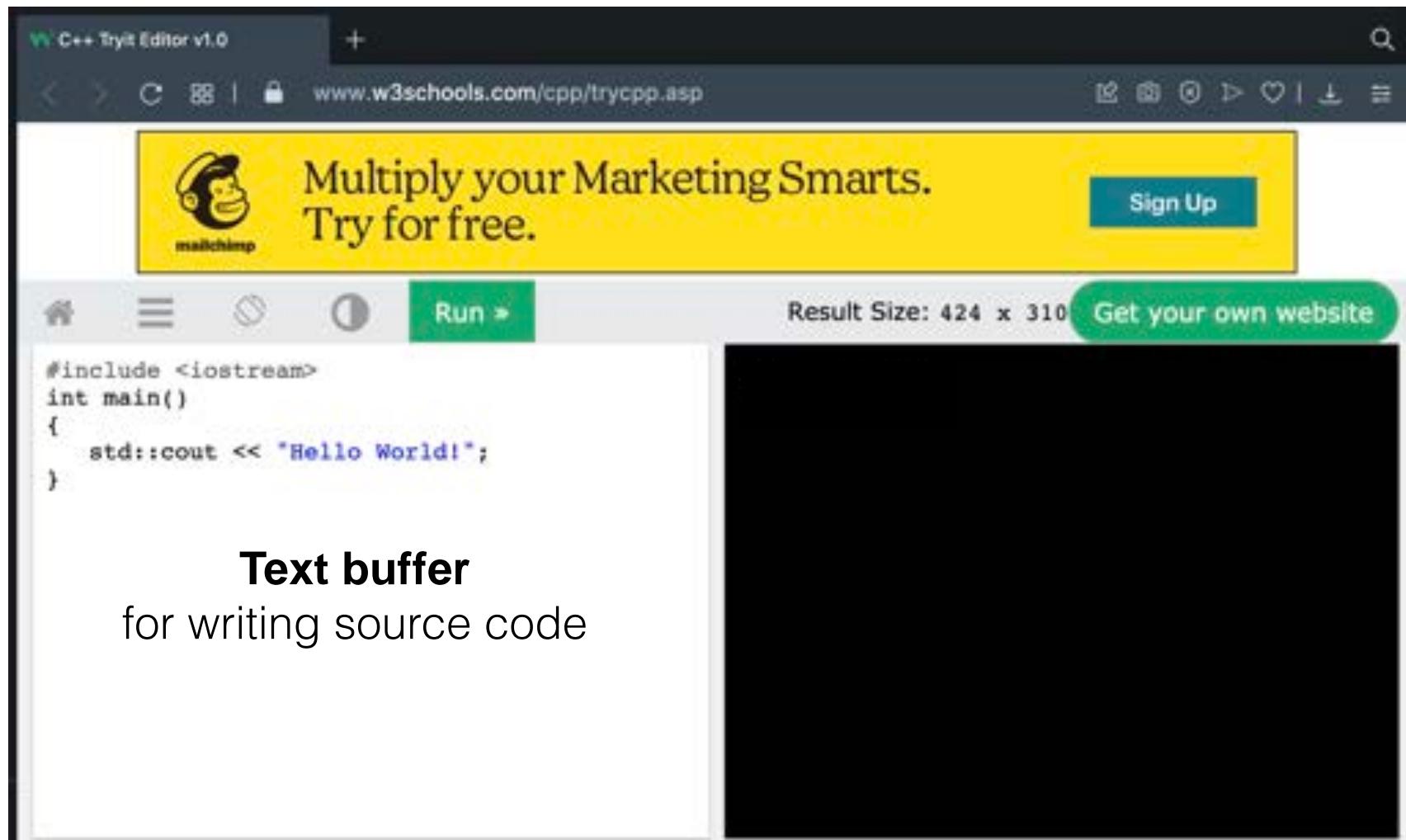
Step 1: Type code into a file

```
#include <iostream>
int main()
{
    std::cout << "Hello World! " ;
}
```

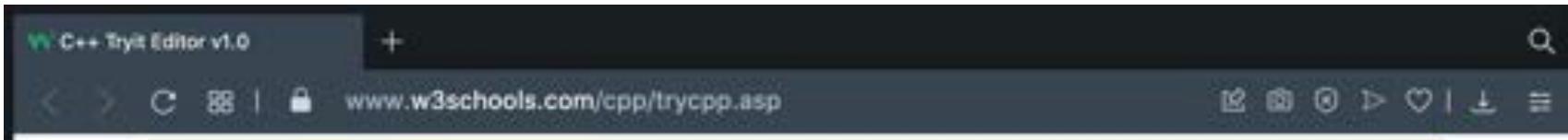
One option: online C++ editors



One option: online C++ editors



One option: online C++ editors



Press “Run”

to compile source code into executable program, and then run the executable

The screenshot shows the C++ Tryit Editor v1.0 interface. On the left, there is a text buffer containing the following C++ code:

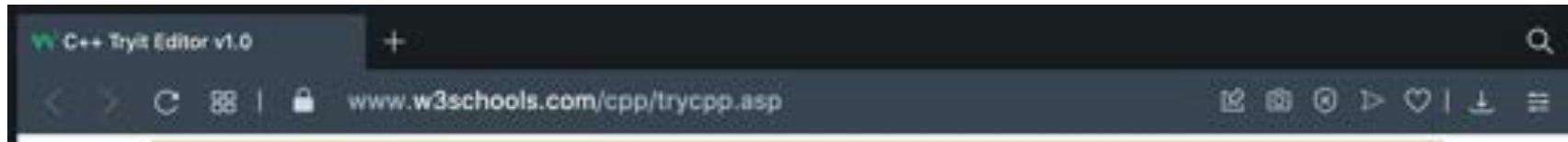
```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

On the right, there is a result window with a black background. Above the result window, the text "Result Size: 424 x 310" and a "Get your own website" button are visible. A red box highlights the "Run >" button, which is located in the toolbar above the text buffer.

Text buffer

for writing source code

One option: online C++ editors



Press “Run”

to compile source code into executable program, and then run the executable

The image shows a split-screen interface for a C++ online editor. On the left, under the heading "Text buffer", there is a code editor containing the following C++ code:

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

On the right, under the heading "Output buffer", there is a terminal-like window showing the output of the program: "Hello World!". A red box highlights the green "Run" button at the top of the editor window.

Text buffer

for writing source code

Output buffer

to see the result of the program

One option: online C++ editors

The screenshot shows a web browser window titled "C++ Tryit Editor v1.0". The address bar indicates the page is "www.w3schools.com/cpp/trycpp.asp". The main content area displays a yellow banner for Mailchimp with the text "Multiply your Marketing Smarts. Try for free." and a "Sign Up" button. Below the banner, there's a toolbar with icons for file operations, a "Run" button, and a status message "Result Size: 424 x 310". A green button says "Get your own website". The code editor contains the following C++ code:

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

The output window shows the result of running the code: "Hello World!". Two boxes are overlaid on the code editor: a red box labeled "Advantages:" and an orange box labeled "Disadvantages:", both containing empty space.

One option: online C++ editors

The screenshot shows a web browser window titled "C++ Tryit Editor v1.0". The address bar indicates the page is at www.w3schools.com/cpp/trycpp.asp. A yellow banner from Mailchimp promotes their marketing services with the text "Multiply your Marketing Smarts. Try for free." and a "Sign Up" button. Below the banner, the code editor displays the following C++ code:

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

The "Run" button is visible above the code area. To the right, the output window shows the result: "Hello World!". The output window has a green header bar with the text "Result Size: 424 x 310" and a "Get your own website" button.

Advantages:

- Easy
- Quick
- Minimal setup

Disadvantages:

Michigan Robotics 102 - robotics102.org

One option: online C++ editors

The screenshot shows a web browser window titled "C++ Tryit Editor v1.0". The address bar displays "www.w3schools.com/cpp/trycpp.asp". The main content area features a yellow banner for Mailchimp with the text "Multiply your Marketing Smarts. Try for free." and a "Sign Up" button. Below the banner, there's a toolbar with icons for file operations, a "Run" button, and a status message "Result Size: 424 x 310". A green button says "Get your own website". The code editor contains the following C++ code:

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

The output window shows the result "Hello World!". The "Advantages" and "Disadvantages" sections are highlighted with red borders.

Advantages:	Disadvantages:
Easy	Advertisements or Pay
Quick	Requires Internet
Minimal setup	Limited use (no robot)

How to code in C++ ?

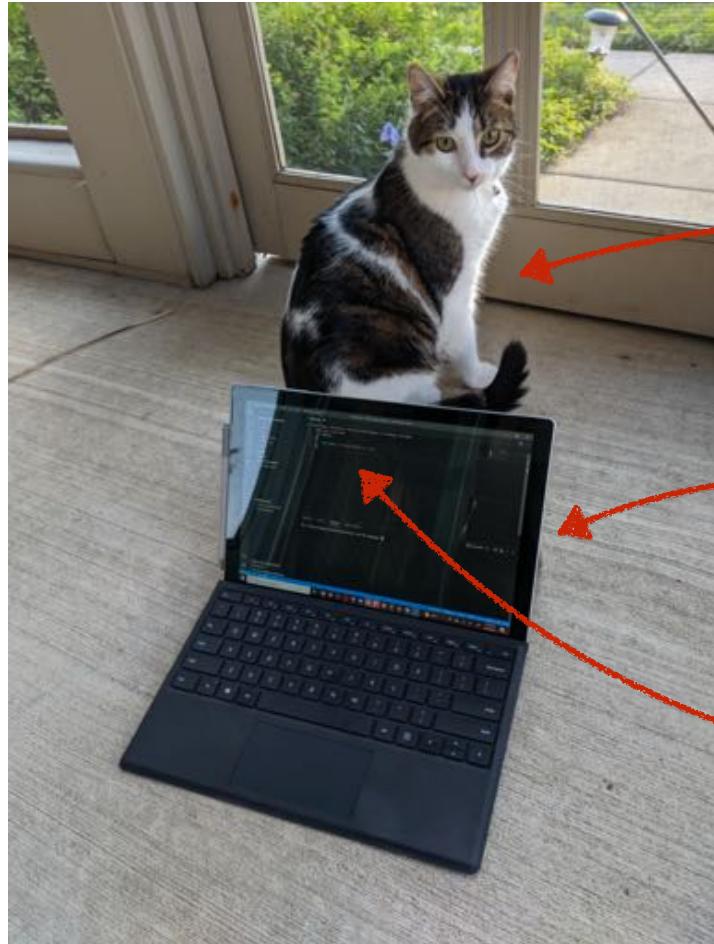
Step 1: Type code into a file

```
#include <iostream>
int main()
{
    std::cout << "Hello World! " ;
}
```

hello.cpp

Michigan Robotics 102 - robotics102.org

Another option: coding in local files



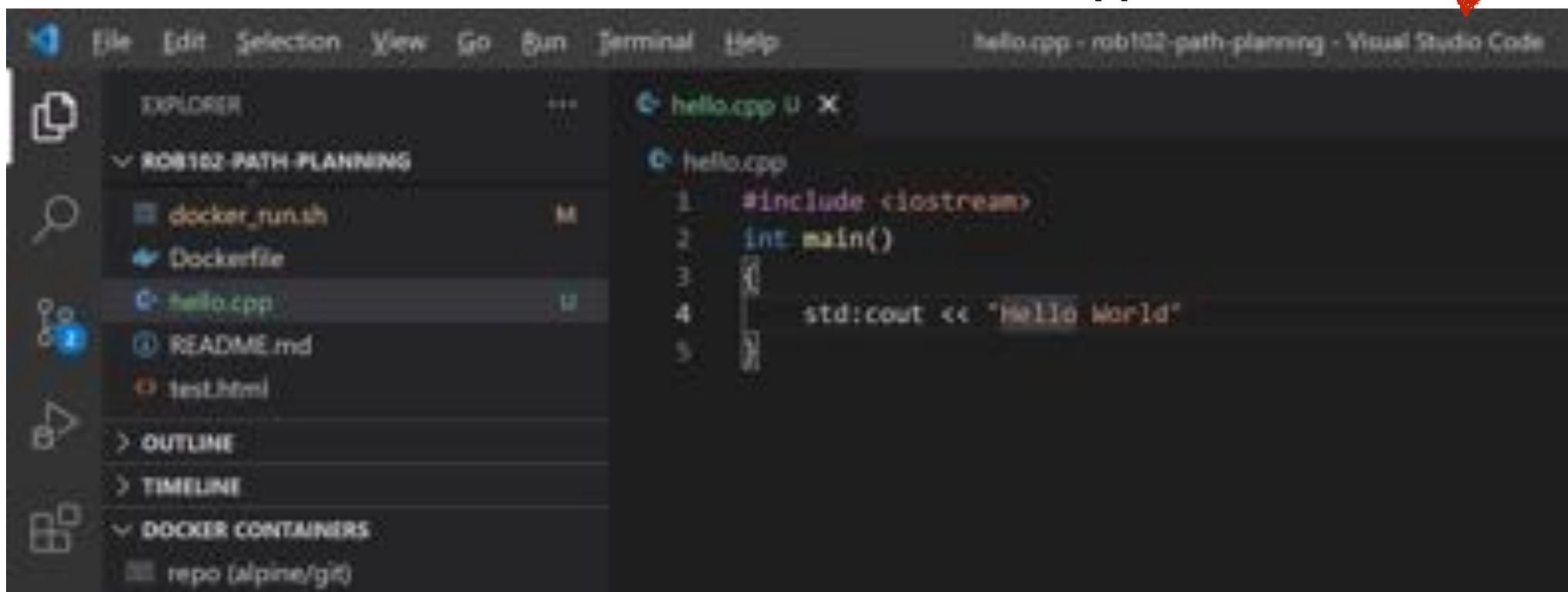
Catari 2600

Microsoft Surface running
Visual Studio Code (VS Code)

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
hello.cpp
```

Step 1: Type code into a file

hello.cpp in VS Code 



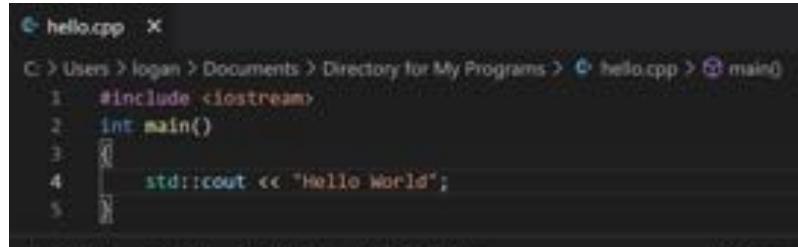
The screenshot shows the Visual Studio Code interface. The title bar reads "hello.cpp - rob102-path-planning - Visual Studio Code". The left sidebar (Explorer) shows a project structure with files: "docker_run.sh", "Dockerfile", "hello.cpp" (which is selected), "README.md", and "test.html". The main editor area displays the following C++ code:

```
#include <iostream>
int main()
{
    std::cout << "Hello World"
}
```

Step 1: Type code into a file

Text editor

Make changes to C++ code



```
hello.cpp  X
C:\Users\logan\Documents\Directory for My Programs> hello.cpp > main()
1 #include <iostream>
2 int main()
3 {
4     std::cout << "Hello World";
5 }
```

hello.cpp

Step 1: Type code into a file

Text editor

Make changes to C++ code

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

hello.cpp

Step 1: Type code into a file

Text editor

Make changes to C++ code

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

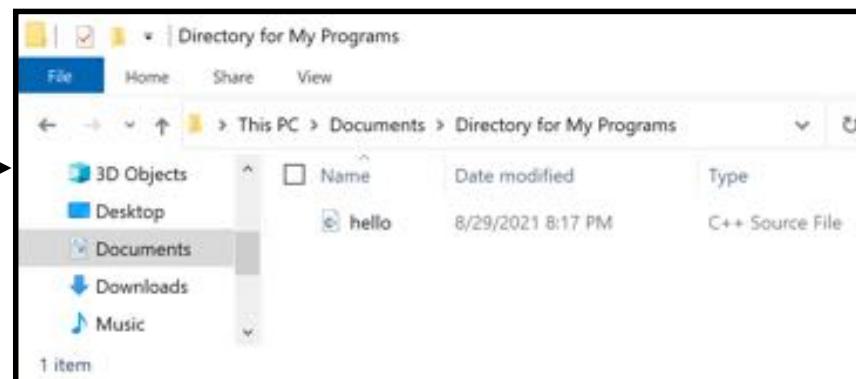
hello.cpp

OPEN FILE

SAVE FILE

Source code

C++ files on computer file system containing your code



Step 1: Type code into a file

Text editor

Make changes to C++ code

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

hello.cpp

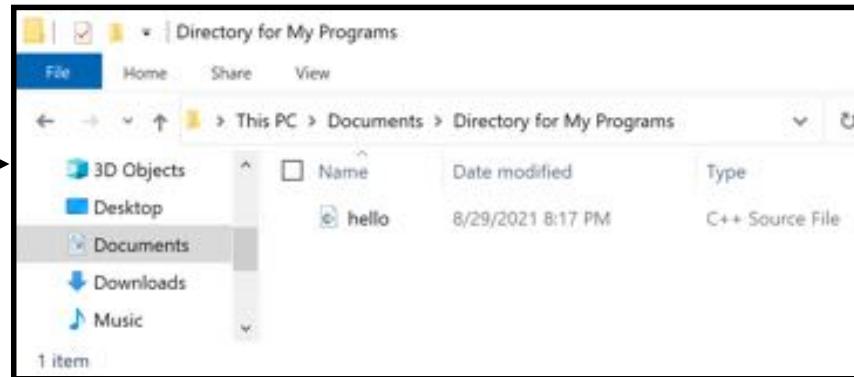
OPEN FILE

SAVE FILE

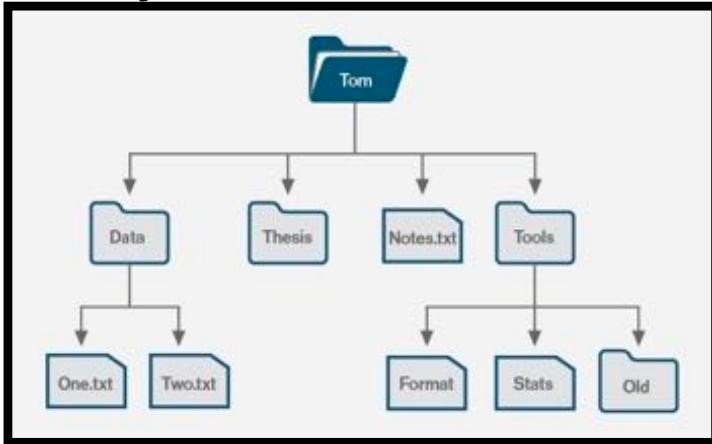
What is a filesystem?

Source code

C++ files on computer file system containing your code

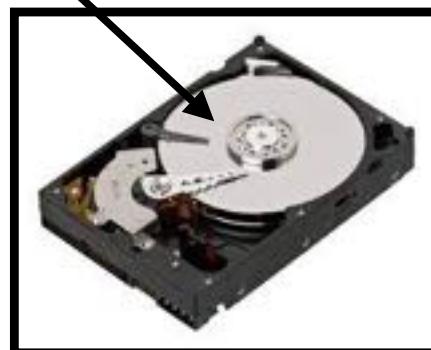


Filesystem

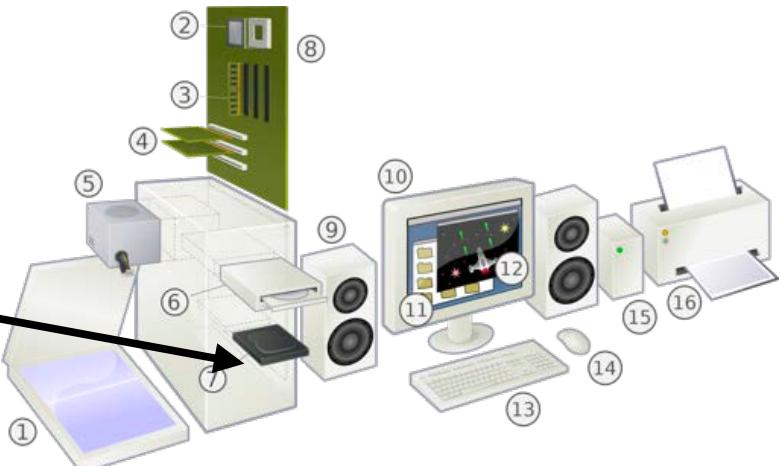


What is a filesystem?

A filesystem organizes information on a storage device into a hierarchy of *directories* that contain data in *files*



Hard Drive Storage



Michigan Robotics 102 - robotics102.org

Step 2: Compile source code

Text editor

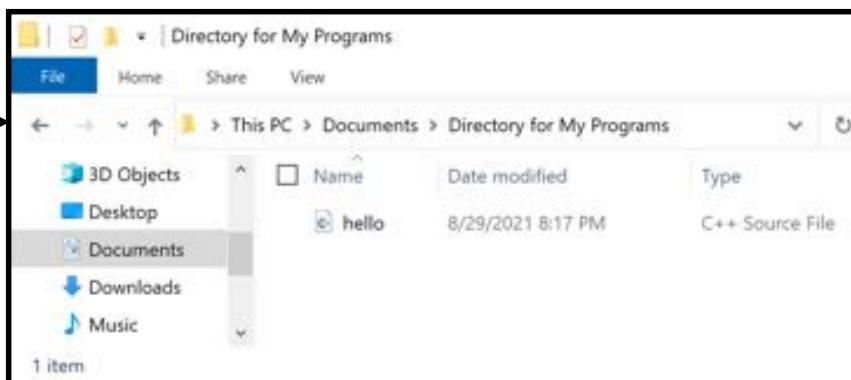
Make changes to C++ code

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

hello.cpp

OPEN FILE

SAVE FILE



Source code

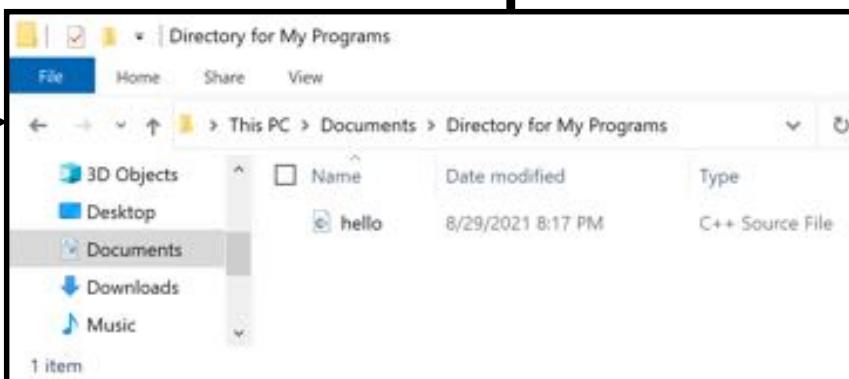
C++ files on computer file system containing your code

Step 2: Compile source code

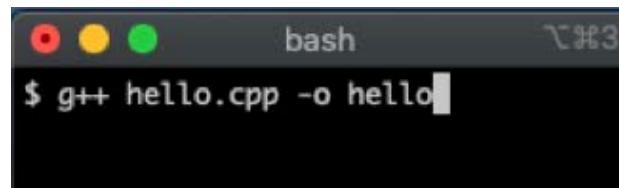
Text editor
Make changes to C++ code

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

hello.cpp
OPEN FILE
SAVE FILE



Compiler
Build executable program
from C++ source files



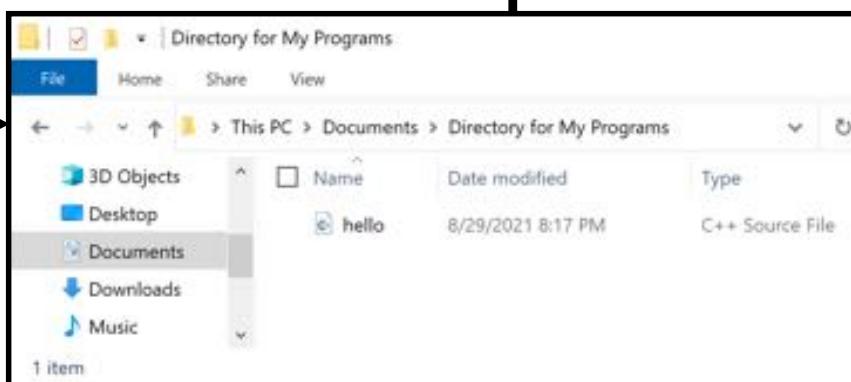
Source code
C++ files on computer file
system containing your code

Step 2: Compile source code

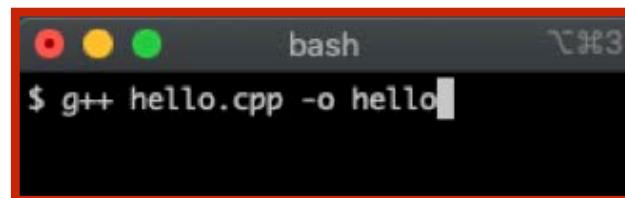
Text editor
Make changes to C++ code

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

hello.cpp
OPEN FILE
SAVE FILE



Compiler
Build executable program
from C++ source files



COMMAND LINE TERMINAL

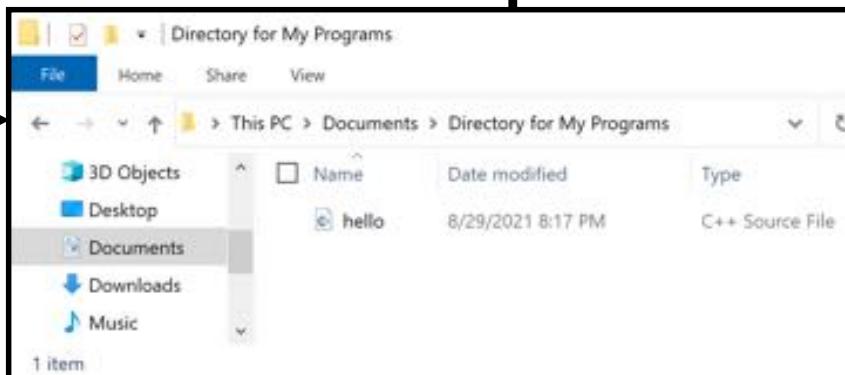
Source code
C++ files on computer file
system containing your code

Step 2: Compile source code

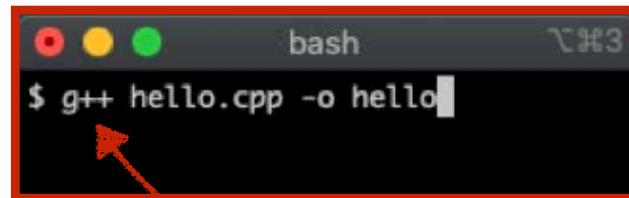
Text editor
Make changes to C++ code

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

hello.cpp
OPEN FILE
SAVE FILE



Compiler
Build executable program
from C++ source files

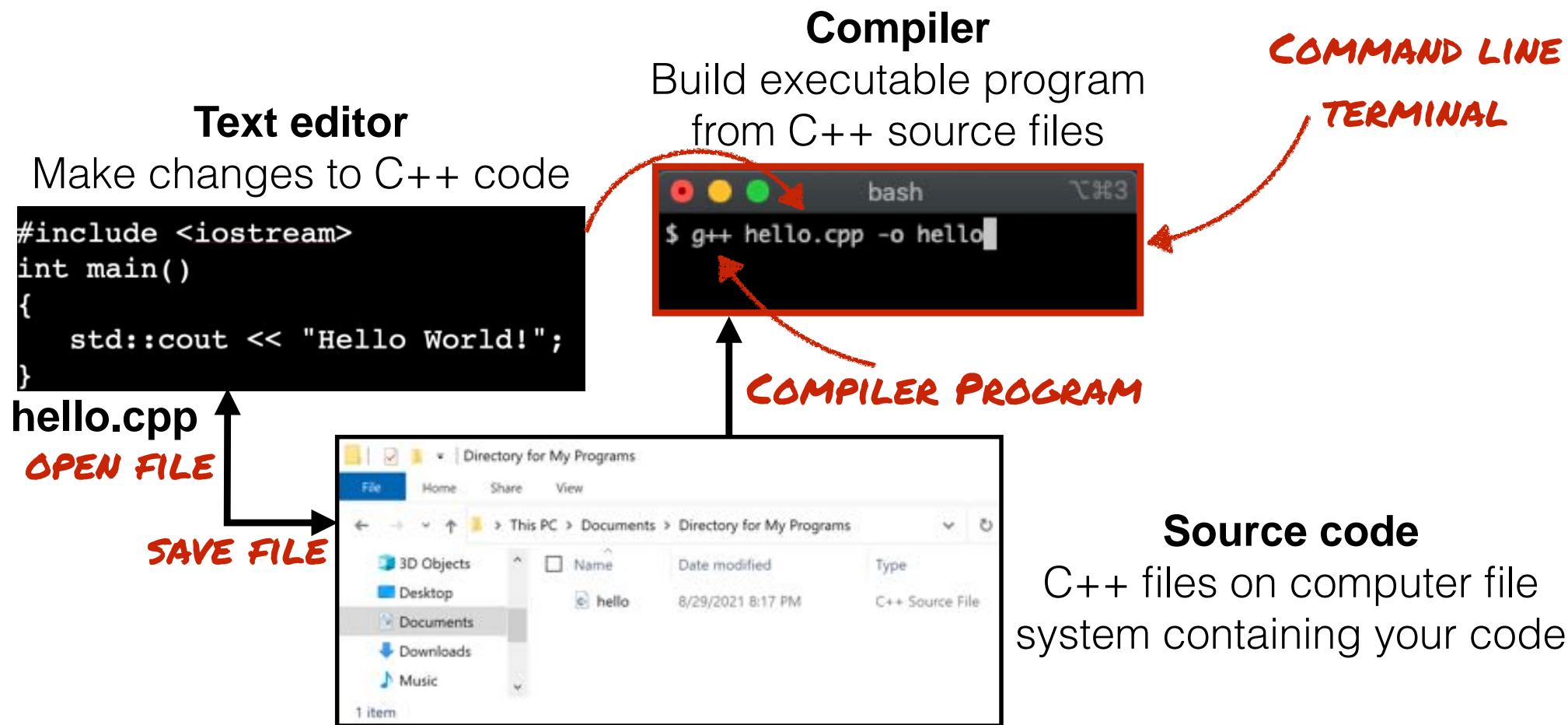


COMPILER PROGRAM

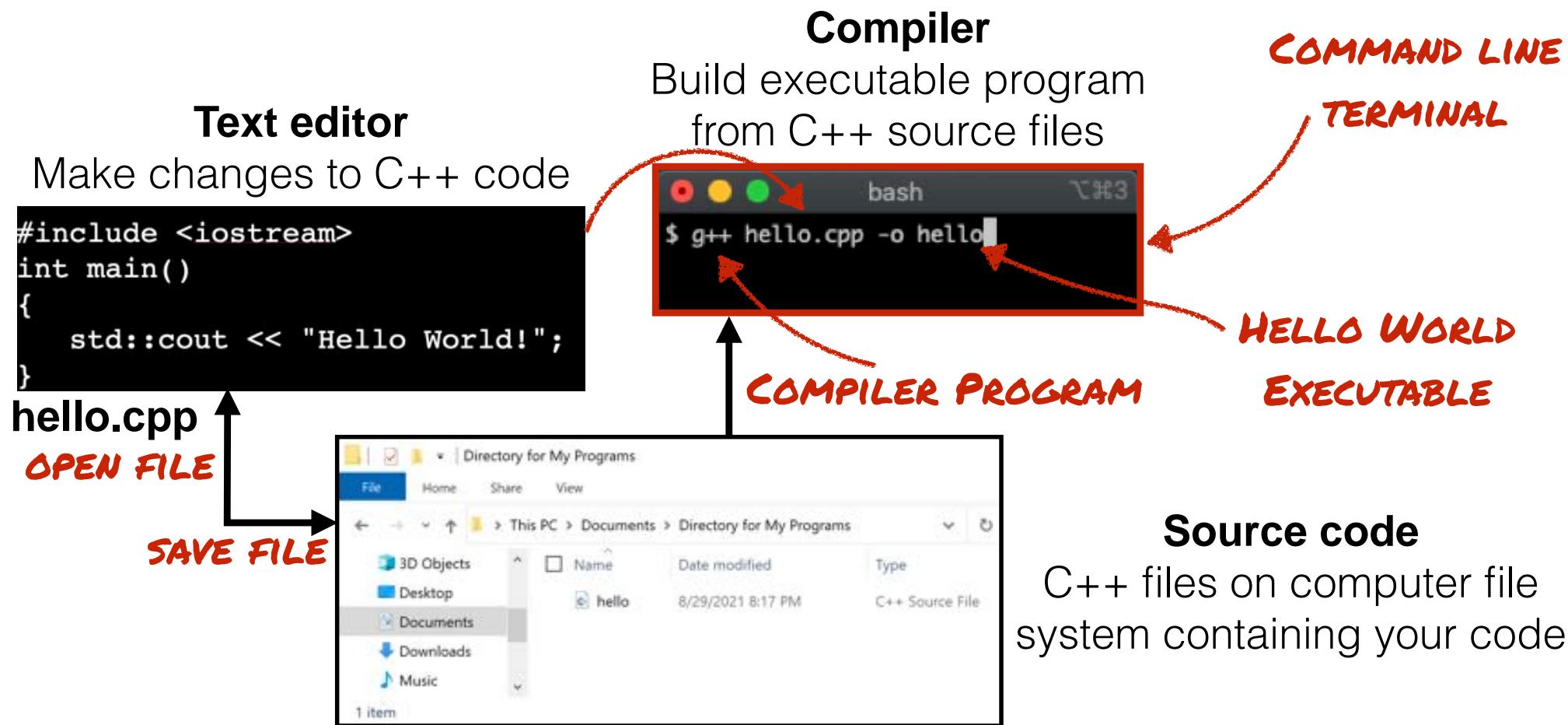
COMMAND LINE TERMINAL

Source code
C++ files on computer file
system containing your code

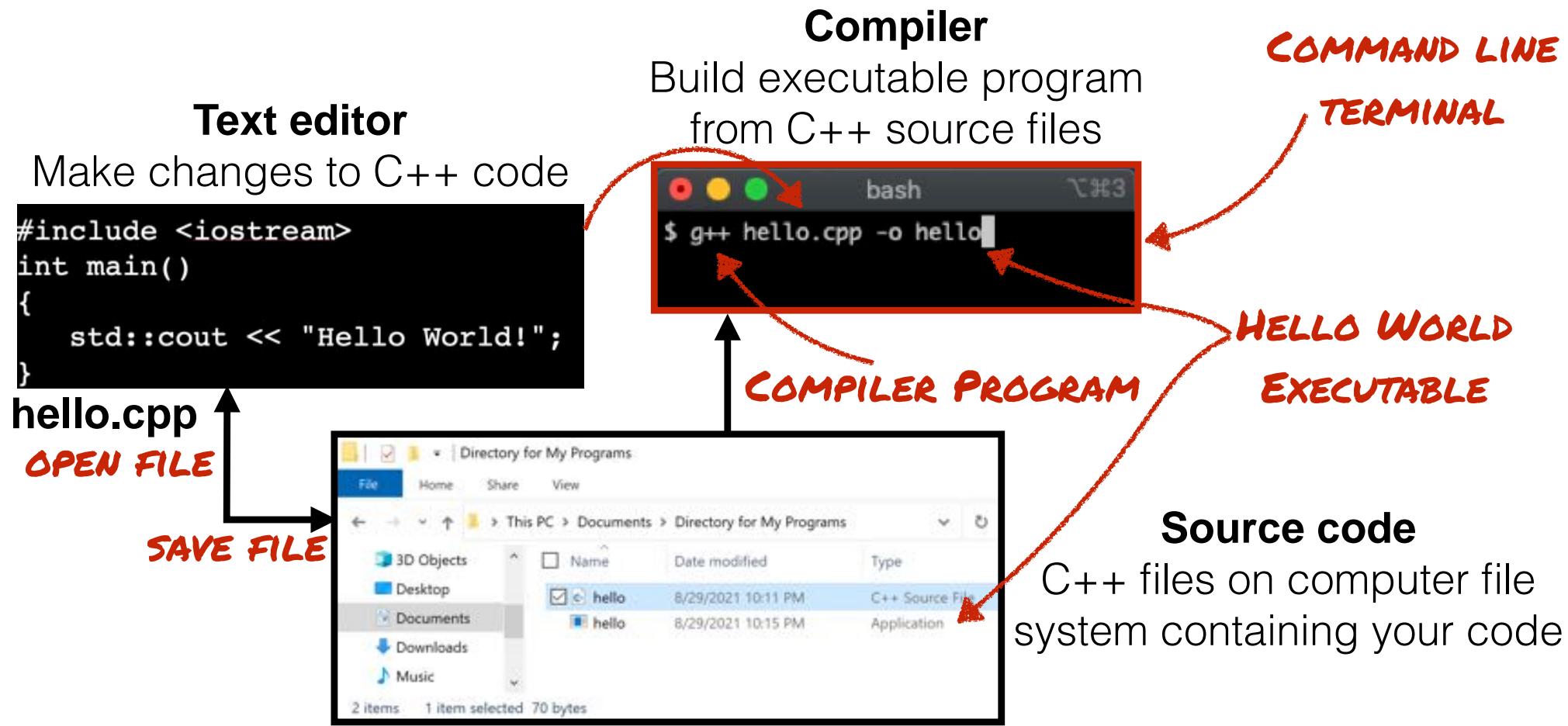
Step 2: Compile source code



Step 2: Compile source code



Step 2: Compile source code

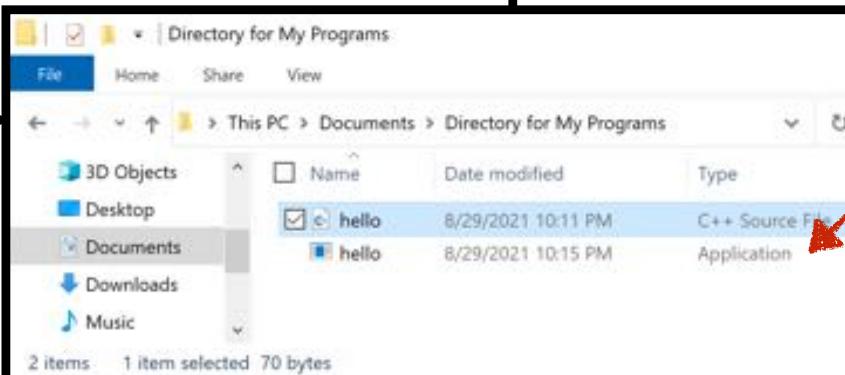


Step 2: Compile source code

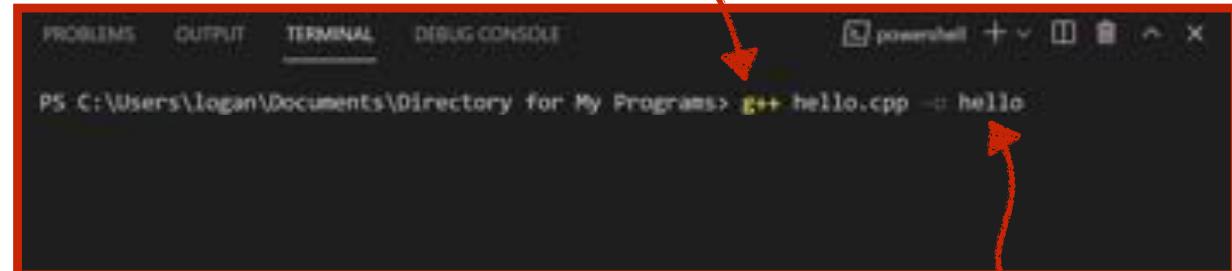
Text editor
Make changes to C++ code

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

hello.cpp
OPEN FILE
SAVE FILE



Compiler
Build executable program
from C++ source files



HELLO WORLD EXECUTABLE
Source code
C++ files on computer file system containing your code

Step 3: Run Executable Program

Text editor

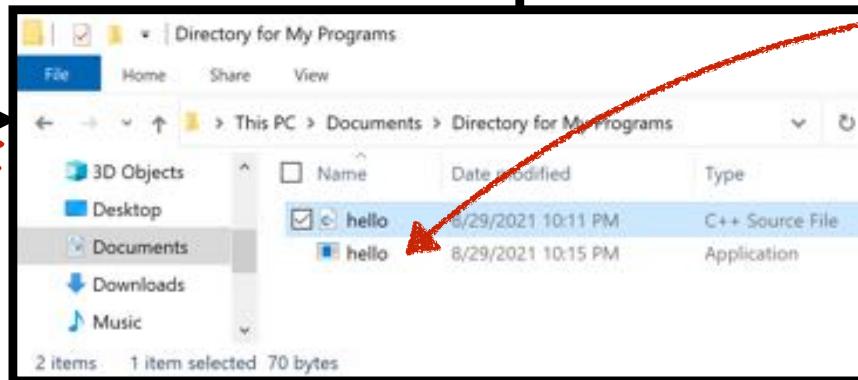
Make changes to C++ code

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

hello.cpp

OPEN FILE

SAVE FILE



Compiler

Build executable program
from C++ source files

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
powershell + x
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
```

Executable

Binary file on computer file
system that can be run

Source code

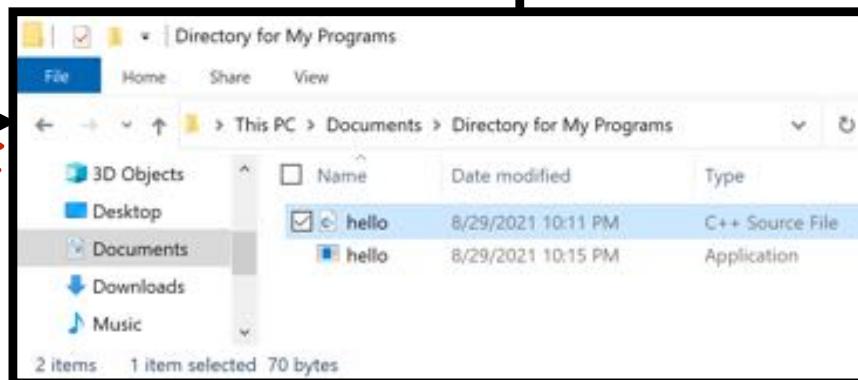
C++ files on computer file
system containing your code

Step 3: Run Executable Program

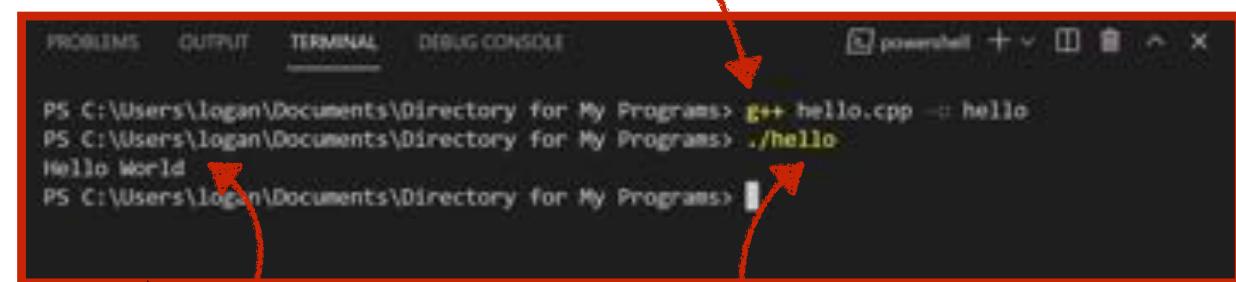
Text editor
Make changes to C++ code

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

hello.cpp
OPEN FILE
SAVE FILE



Compiler
Build executable program
from C++ source files



PROGRAM OUTPUT

Executable
Binary file on computer file
system that can be run

Source code
C++ files on computer file
system containing your code

Compiler Executable

```
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello World
PS C:\Users\logan\Documents\Directory for My Programs>
```

Text editor

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

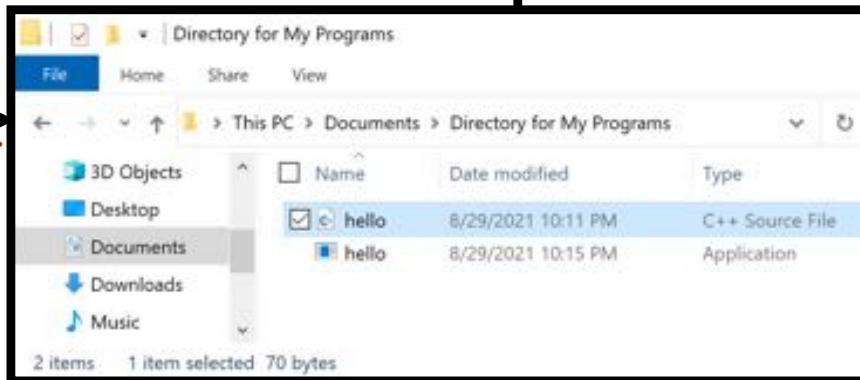
hello.cpp

OPEN FILE

SAVE FILE

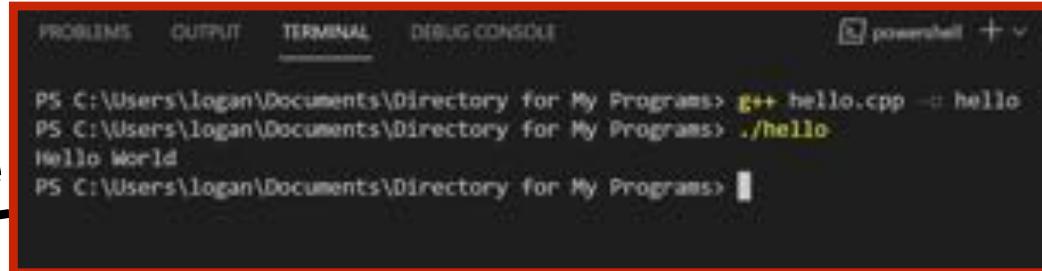
Source code

**COMPILE
AND EXECUTE**



Compiler
Executable

IDEAS



```
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello World
PS C:\Users\logan\Documents\Directory for My Programs>
```

Text editor

```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

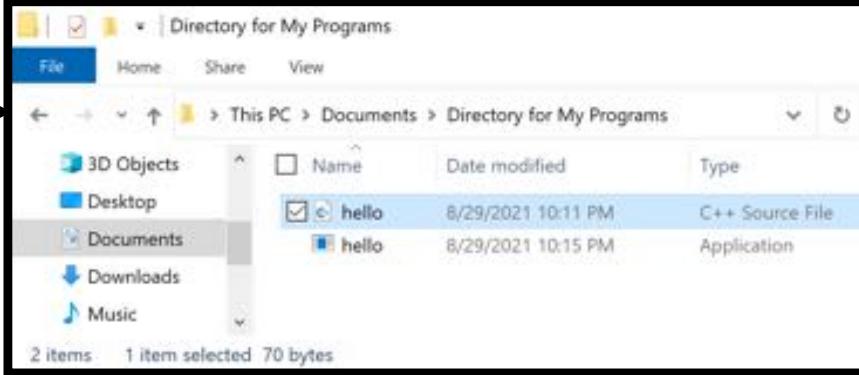
hello.cpp

OPEN FILE

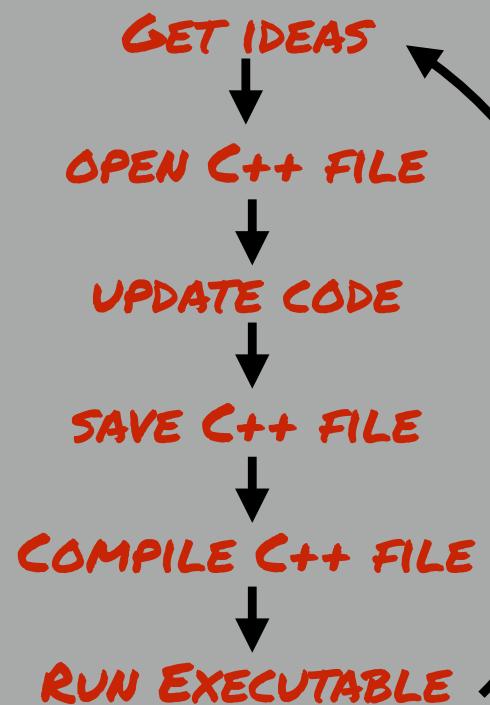
SAVE FILE

Source code

**COMPILE
AND EXECUTE**



Coding process



Compiler
Executable
IDEAS

Text editor

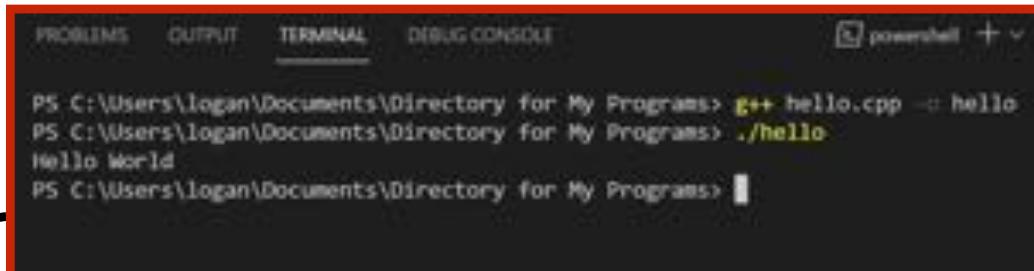
```
#include <iostream>
int main()
{
    std::cout << "Hello World!";
}
```

hello.cpp

OPEN FILE

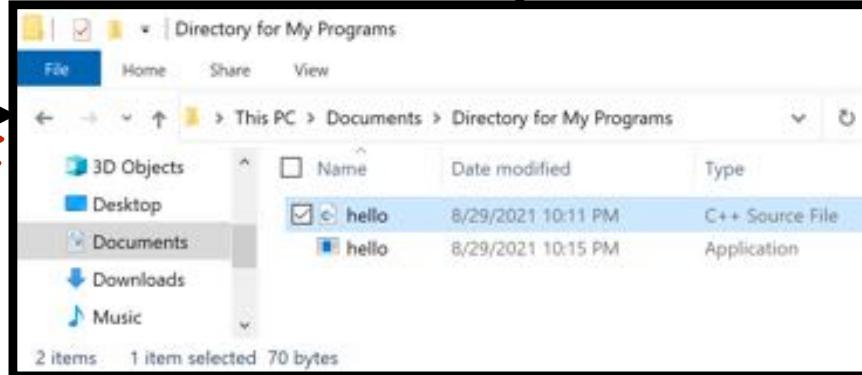
SAVE FILE

Source code

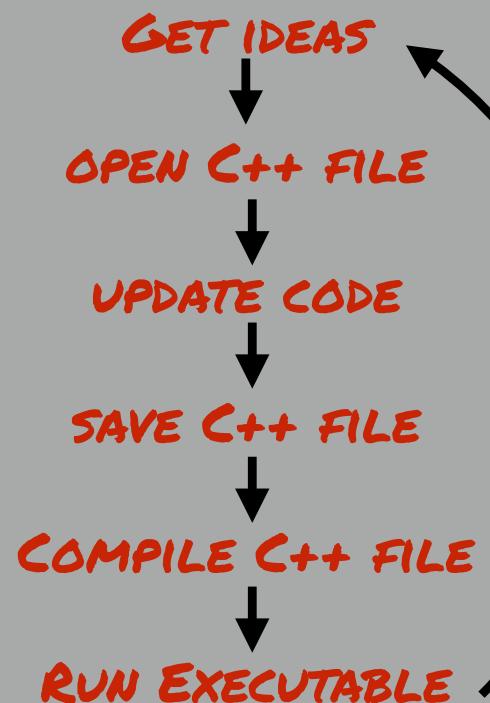


```
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello World
PS C:\Users\logan\Documents\Directory for My Programs>
```

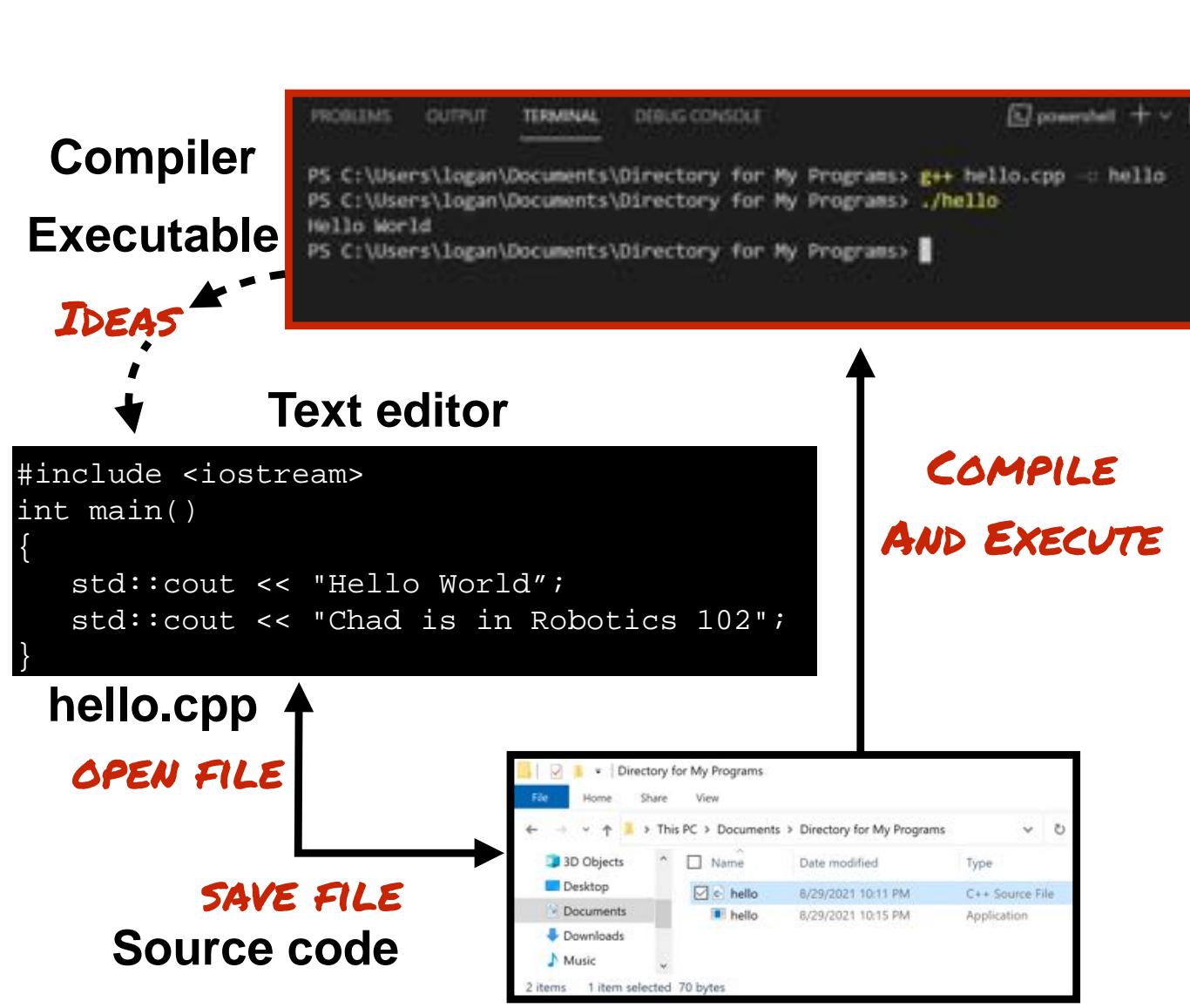
**COMPILE
AND EXECUTE**



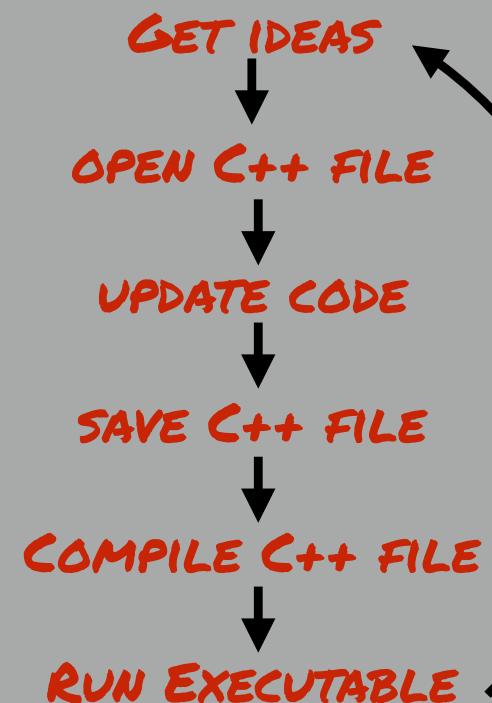
Coding process



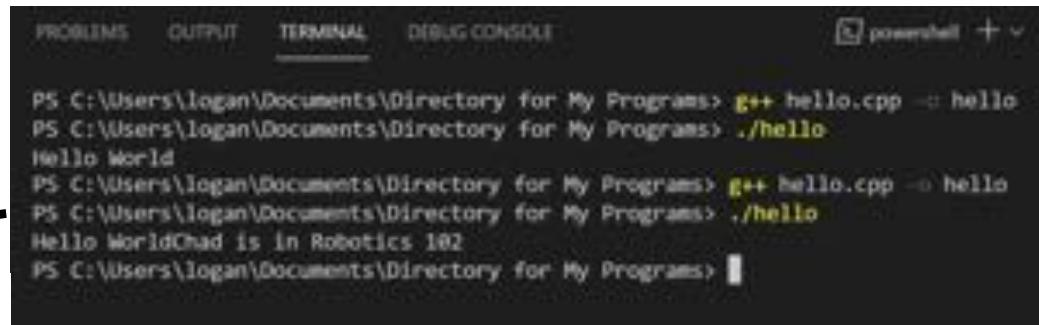
New idea:
Print that I am in 102



Coding process



Compiler
Executable
IDEAS



```
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello World
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello WorldChad is in Robotics 102
PS C:\Users\logan\Documents\Directory for My Programs>
```

Text editor

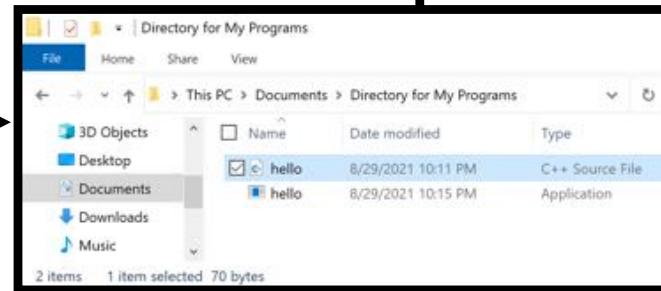
```
#include <iostream>
int main()
{
    std::cout << "Hello World";
    std::cout << "Chad is in Robotics 102";
}
```

hello.cpp

OPEN FILE

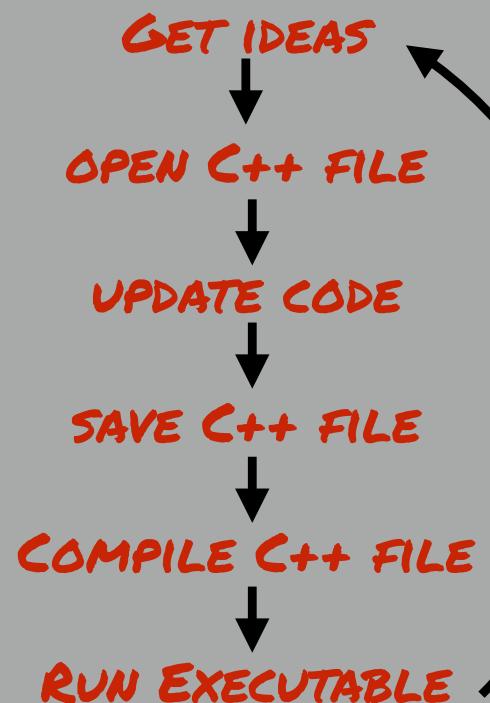
SAVE FILE

Source code

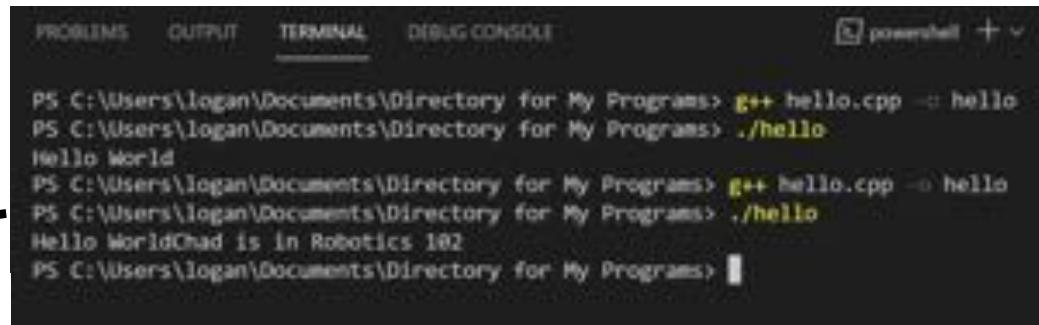


**COMPILE
AND EXECUTE**

Coding process



Compiler
Executable
IDEAS



```
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello World
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello WorldChad is in Robotics 102
PS C:\Users\logan\Documents\Directory for My Programs>
```

Text editor

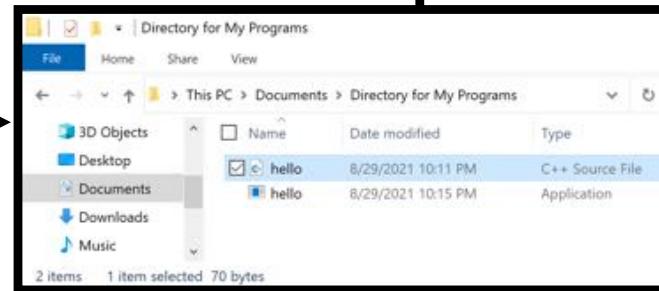
```
#include <iostream>
int main()
{
    std::cout << "Hello World";
    std::cout << "Chad is in Robotics 102";
}
```

hello.cpp

OPEN FILE

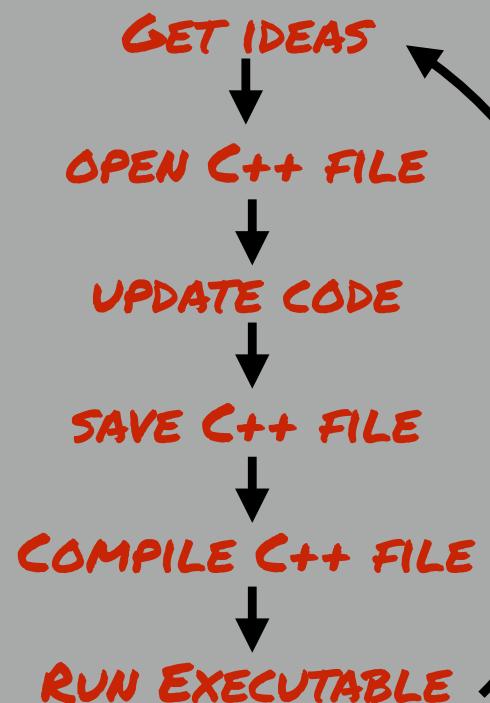
SAVE FILE

Source code



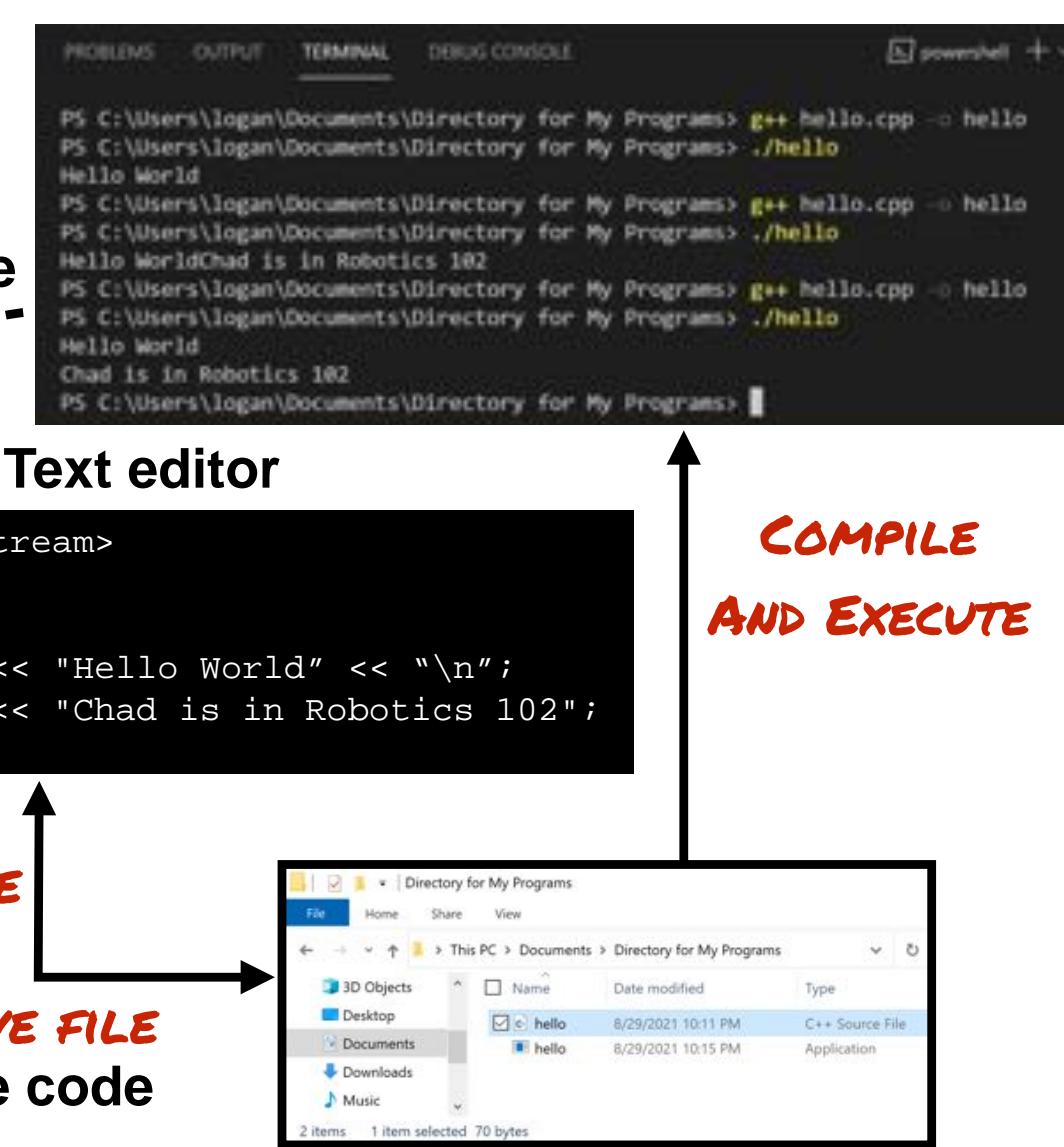
**COMPILE
AND EXECUTE**

Coding process

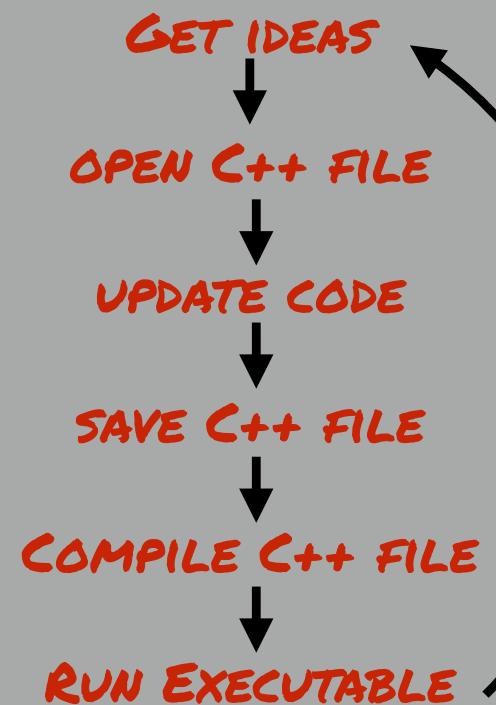


**New idea:
Break each line**

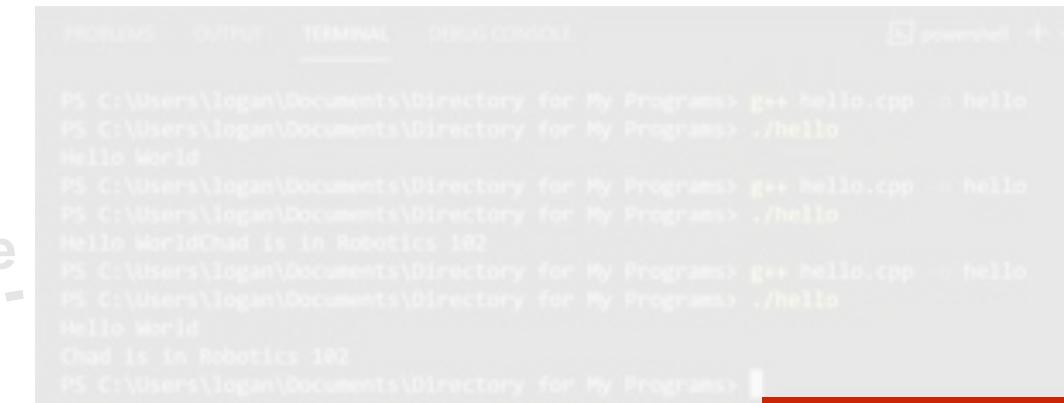
Compiler
Executable
IDEAS



Coding process



Compiler
Executable
IDEAS



```
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello World
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello WorldChad is in Robotics 102
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello World
Chad is in Robotics 102
PS C:\Users\logan\Documents\Directory for My Programs>
```

Text editor

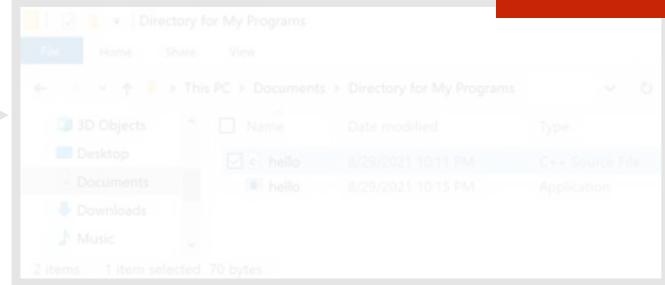
```
#include <iostream>
int main()
{
    std::cout << "Hello World" << "\n";
    std::cout << "Chad is in Robotics 102";
}
```

hello.cpp

OPEN FILE

SAVE FILE

Source code



Coding process

GET IDEAS

OPEN C++ FILE

Wait

AND EXECUTE

You did what?

Anatomy of a C++ Program

```
#include <iostream>
int main( )
{
    std::cout << "Hello World" << "\n";
    std::cout << "Chad is in Robotics 102";
}
```

hello.cpp

Anatomy of a C++ Program

***All programs start in
the main function***



```
#include <iostream>
int main( )
{
    std::cout << "Hello World" << "\n";
    std::cout << "Chad is in Robotics 102";
}
```

hello.cpp

Anatomy of a C++ Program

**All programs start in
the main function**



```
#include <iostream>
int main( )
{
    std::cout << "Hello World" << "\n";
    std::cout << "Chad is in Robotics 102";
}
```

hello.cpp

**Scope of main function
delimited by matching braces**

Anatomy of a C++ Program

**All programs start in
the main function**

The diagram shows a C++ code snippet in a black box. A red arrow points from the text "All programs start in the main function" to the opening brace of the main function. Another red arrow points from the text "Scope of main function delimited by matching braces" to the closing brace of the main function. Inside the main function, two semicolons at the end of cout statements are highlighted with red boxes, and red arrows point from the text "Statements are program instructions and end with a semicolon" to them.

```
#include <iostream>
int main()
{
    std::cout << "Hello World" << "\n";
    std::cout << "Chad is in Robotics 102";
}
```

hello.cpp

**Scope of main function
delimited by matching braces**

**Statements are program instructions
and end with a semicolon**

**All programs start in
the main function**

The diagram shows a C++ code snippet named `hello.cpp`. It includes the `#include <iostream>` directive, the `int main()` function definition, and two `std::cout` statements. The first `cout` statement prints "Hello World" followed by a new line character. The second `cout` statement prints "Chad is in Robotics 102". Red annotations highlight the `main()` function, the brace pair {}, and the two `cout` statements. A red arrow points from the `main()` annotation to the opening brace {}, and another arrow points from the brace {} to the closing brace }. A red circle highlights the entire `main()` function body. A red arrow points from the `cout` annotation to the first `cout` statement, and another arrow points from the first `cout` statement to the second.

```
#include <iostream>
int main()
{
    std::cout << "Hello World" << "\n";
    std::cout << "Chad is in Robotics 102";
}
```

hello.cpp

**Scope of main function
delimited by matching braces**

**First statement prints to
“standard output” (the screen)
the string (in quotes) followed by
a new line character**

**Statements are program instructions
and end with a semicolon**

**All programs start in
the main function**

The diagram shows a code editor window with the file 'hello.cpp' containing C++ code. Red arrows and boxes highlight specific parts of the code:

- A large red arrow points from the text "All programs start in the main function" to the opening brace of the main function definition.
- Red boxes highlight the following code blocks:
 - The entire main function definition: `#include <iostream>`, `int main()`, opening brace, two `std::cout <<` statements, and closing brace.
 - The two `std::cout <<` statements, each followed by its string argument ("Hello World" and "Chad is in Robotics 102") and a semicolon.
 - The semicolons at the end of each `std::cout <<` statement and the final closing brace of the main function.
- A red box highlights the text "Insertion ("put") operator".
- A red arrow points from the text "First statement prints to standard output (the screen) the string (in quotes) followed by a new line character" to the first `std::cout <<` statement.

```
#include <iostream>
int main()
{
    std::cout << "Hello World" << "\n";
    std::cout << "Chad is in Robotics 102";
}
```

**Scope of main function
delimited by matching braces**

**First statement prints to
"standard output" (the screen)
the string (in quotes) followed by
a new line character**

**Insertion ("put")
operator**

**Statements are program instructions
and end with a semicolon**

**Include the
C++ Input-Output Stream Library,
which provides std::cout**

**All programs start in
the main function**

```
#include <iostream>
int main( )
{
    std::cout << "Hello World" << "\n";
    std::cout << "Chad is in Robotics 102";
}
```

The code is annotated with several red boxes and arrows:

- A large red circle surrounds the entire code block.
- A red box highlights the line `#include <iostream>`.
- A red box highlights the opening brace of the `main` function: `{`.
- A red box highlights the closing brace of the `main` function: `}`.
- A red box highlights the first `std::cout` statement: `std::cout << "Hello World" << "\n";`.
- A red box highlights the second `std::cout` statement: `std::cout << "Chad is in Robotics 102";`.
- A red box highlights the insertion operator `<<`.
- A red box highlights the string literal `"Hello World"`.
- A red box highlights the string literal `"Chad is in Robotics 102"`.
- A red box highlights the new line character `\n`.
- A red box highlights the closing brace of the `main` function: `;`.
- A red box highlights the semicolon at the end of the second `std::cout` statement: `;`.
- A red arrow points from the `#include` line to the `std::cout` statements.
- A red arrow points from the opening brace of `main` to the `std::cout` statements.
- A red arrow points from the closing brace of `main` to the `std::cout` statements.
- A red arrow points from the `<<` operator to the `"Hello World"` string.
- A red arrow points from the `<<` operator to the `"\n"`.
- A red arrow points from the `<<` operator to the `"Chad is in Robotics 102"` string.
- A red arrow points from the `;"` at the end of the second `std::cout` statement back to the closing brace of `main`.

**Scope of main function
delimited by matching braces**

**First statement prints to
“standard output” (the screen)
the string (in quotes) followed by
a new line character**

**Insertion (“put”)
operator**

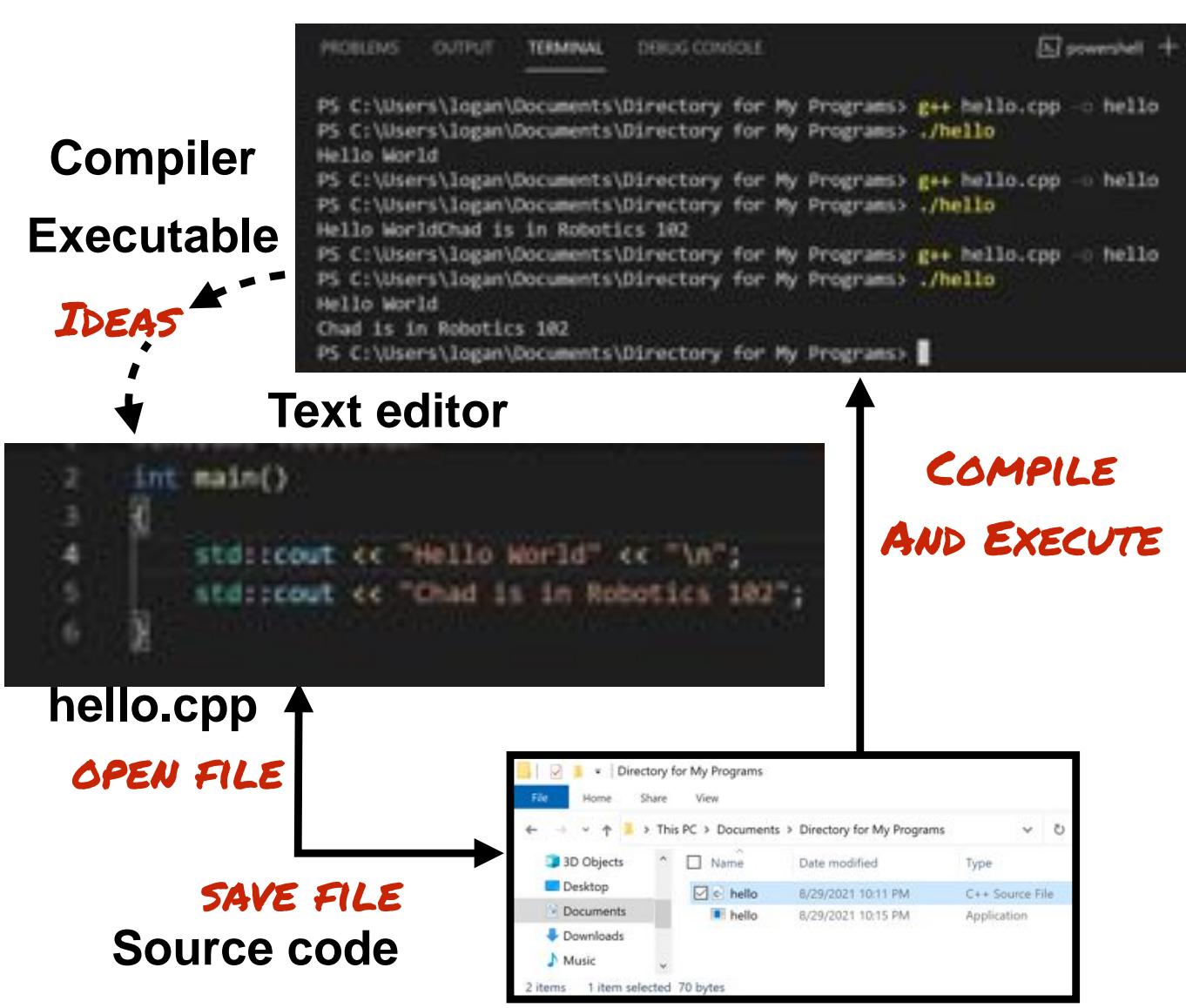
**Statements are program instructions
and end with a semicolon**

Statements are executed in sequential order based on where they appear in the program

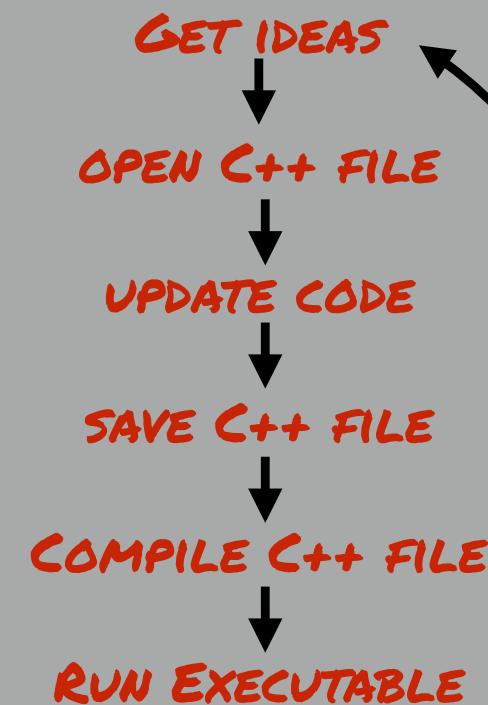
```
#include <iostream>
int main()
{
    std::cout << "Hello World" << "\n";
    std::cout << "Chad is in Robotics 102";
}
```

hello.cpp

This statement is executed after this statement



Coding process



VS Code is an Integrated Development Environment (or IDE)

The screenshot shows the Visual Studio Code interface with the following components labeled:

- FILE EXPLORER**: Located on the left side, showing a tree view of the project structure under "ROB102-PATH-PLANNING".
- SOURCE FILE**: The main editor area showing a C++ file named "hello.cpp". The code prints "Hello World" and "Chad is in Robotics 102".
- TERMINAL**: The bottom pane showing a terminal window with command-line output from running the program.

Terminal Output:

```
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello World
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello WorldChad is in Robotics 102
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello World
Chad is in Robotics 102
PS C:\Users\logan\Documents\Directory for My Programs>
```

Terminal Help

hello.cpp - rob102-path-planning - Visual Studio Code

C:\hello.cpp X

C:\Users\logan\Documents\Directory for My Programs> C:\hello.cpp > main()

```
2 int main()
3 {
4     std::cout << "Hello World" << "\n";
5     std::cout << "Chad is in Robotics 102";
6 }
```

SOURCE FILE

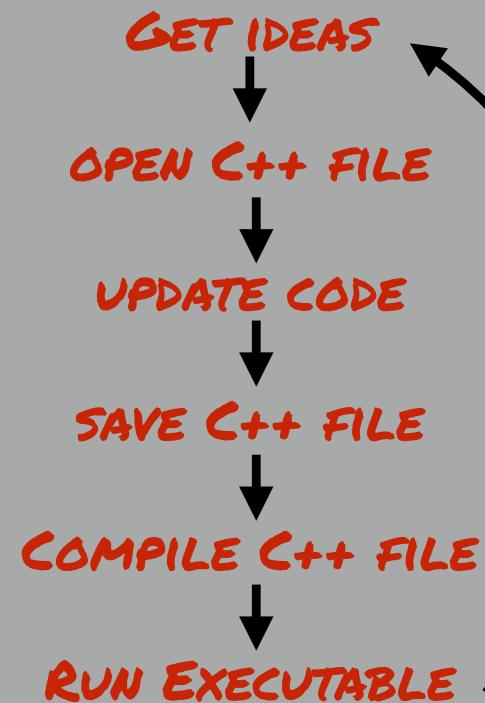
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

powerShell + v

```
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello COMPILE
PS C:\Users\logan\Documents\Directory for My Programs> ./hello EXECUTE
Hello World
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello COMPILE
PS C:\Users\logan\Documents\Directory for My Programs> ./hello EXECUTE
Hello WorldChad is in Robotics 102
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello COMPILE
PS C:\Users\logan\Documents\Directory for My Programs> ./hello EXECUTE
Hello World
Chad is in Robotics 102
PS C:\Users\logan\Documents\Directory for My Programs>
```

Line: 4 Col: 38 Spaces: 4 UTF-8 CRLF C++ Win32 RP D

Coding process



What happens if I make a mistake?

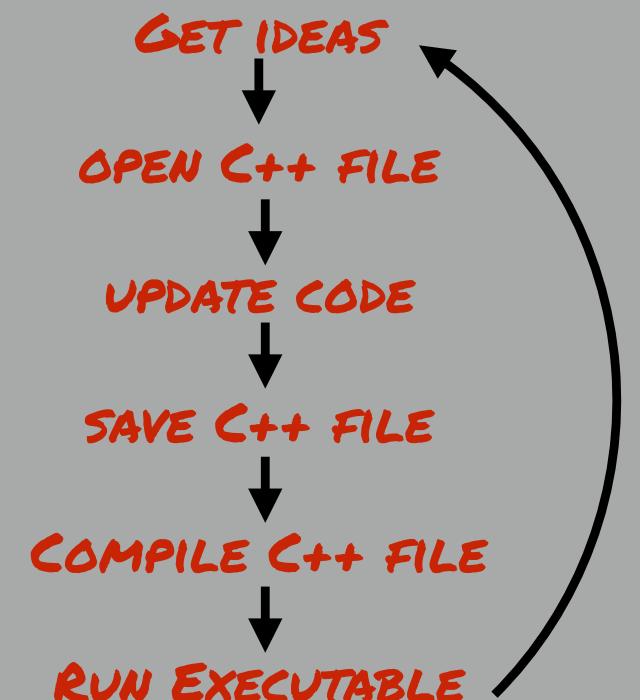
The screenshot shows a Visual Studio Code interface. The top part is a code editor with a dark theme, displaying a C++ source file named `hello.cpp`. The code prints "Hello World" and "Chad is in Robotics 102". The bottom part is a terminal window showing the execution of the program. The terminal output is as follows:

```
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello COMPILE
PS C:\Users\logan\Documents\Directory for My Programs> ./hello EXECUTE
Hello World
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello COMPILE
PS C:\Users\logan\Documents\Directory for My Programs> ./hello EXECUTE
Hello WorldChad is in Robotics 102
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello COMPILE
PS C:\Users\logan\Documents\Directory for My Programs> ./hello EXECUTE
Hello World
Chad is in Robotics 102
PS C:\Users\logan\Documents\Directory for My Programs>
```

Annotations in white text have been added to the terminal output:

- "COMPILE" is written next to the first two `g++` commands.
- "EXECUTE" is written next to the first two `./hello` commands.
- "COMPILE" is written next to the third `g++` command.
- "EXECUTE" is written next to the third `./hello` command.

Coding process

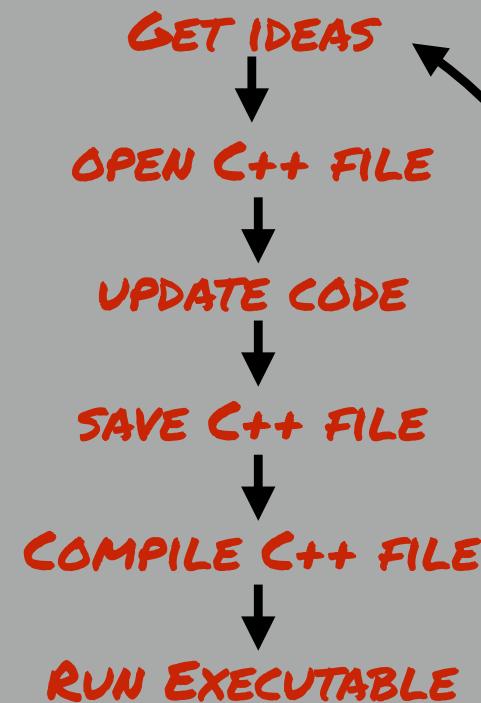


Suppose a semicolon is forgotten

A screenshot of Visual Studio Code showing a C++ file named `hello.cpp`. The code contains a `main()` function with two `std::cout` statements. The second statement lacks a final semicolon, resulting in a syntax error. A red rectangular box covers the bottom right portion of the code editor, containing the word "Oops" in white.

```
C:\Users\logan>Documents>Directory for My Programs>C:\hello.cpp>main()
1 int main()
2 {
3     std::cout << "Hello World" << "\n";
4     std::cout << "Chad is in Robotics 182"
5 }
```

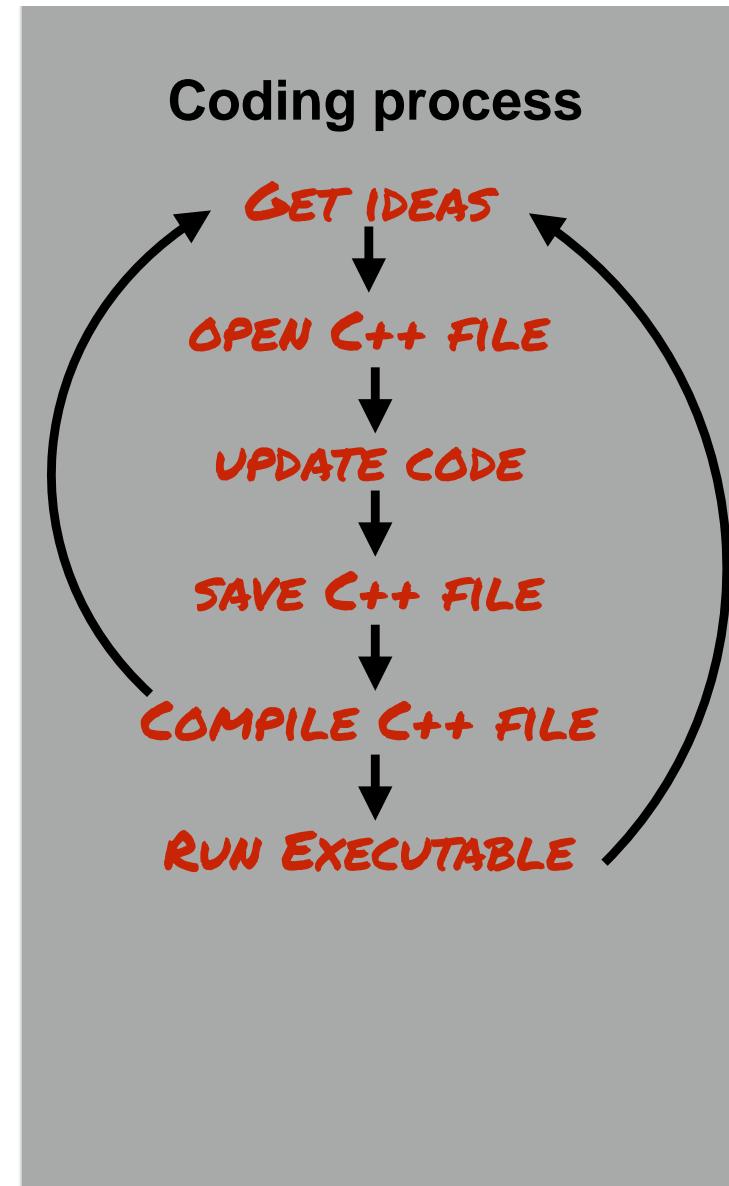
Coding process



Suppose a semicolon is forgotten

A screenshot of Visual Studio Code showing a C++ file named `hello.cpp`. The code contains a `main` function with two `std::cout` statements. The last line of code, which should be `5 std::cout << "Chad is in Robotics 102";`, is missing its final semicolon. A large orange rectangular overlay with the word "Oops" in white is centered over the code editor. Below the editor, the terminal window shows the compilation command `g++ hello.cpp -o hello` followed by the error message: `hello.cpp: In function 'int main()': hello.cpp:5:43: error: expected ';' before ')' token`. The status bar at the bottom indicates the code is at line 5, column 43.

Compilation will fail with an error



Suppose a scoping brace is forgotten

A screenshot of the Visual Studio Code interface. The title bar says "hello.cpp - rob102-path-planning - Visual Studio Code". The code editor shows the following C++ code:

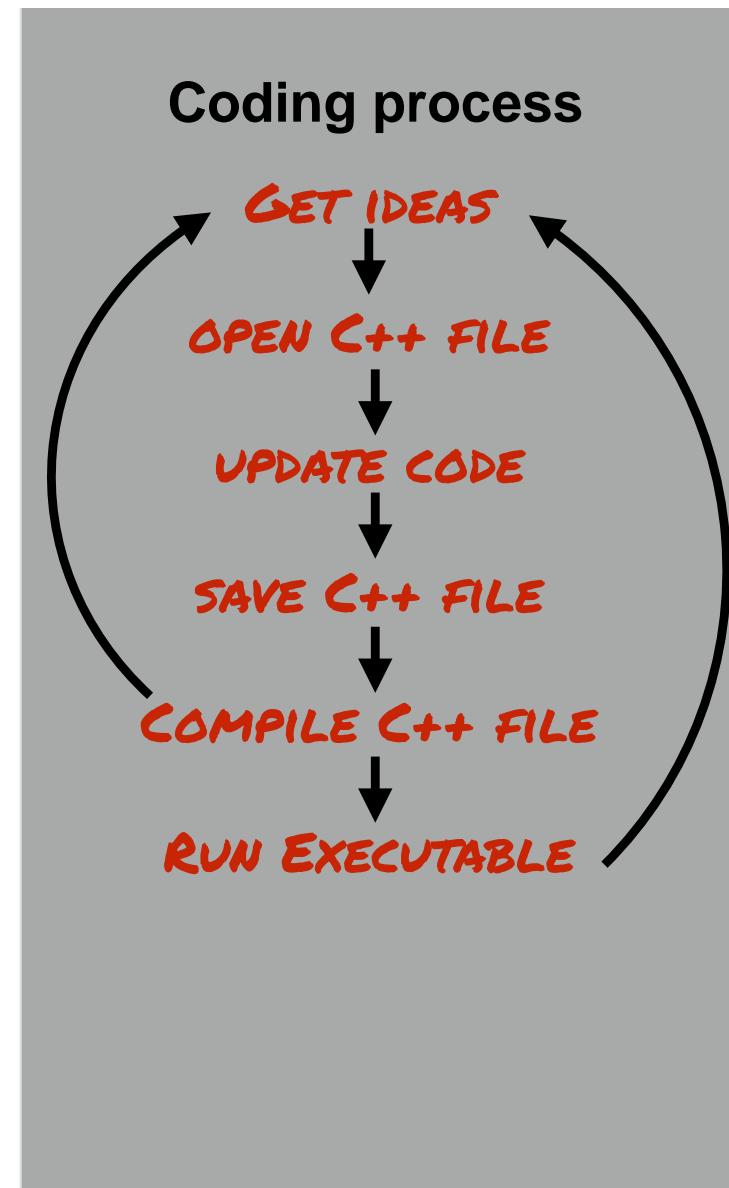
```
1 int main()
2 {
3     std::cout << "Hello World" << "\n";
4     std::cout << "Chad is in Robotics 102";
5 }
```

The closing brace at line 5 is highlighted with a red box. A large orange "Oops" button is overlaid on the center of the code editor. Below the code editor is a terminal window showing the command line output of a compilation attempt:

```
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp --o hello
hello.cpp: In function 'int main()':
hello.cpp:5:43: error: expected ')' at end of input
      5 |     std::cout << "Chad is in Robotics 102";
          |
hello.cpp:3:1: note: to match this '('
  3 | 
```

The status bar at the bottom of the screen shows "LH 6, Col 1" and other file-related information.

Compilation will fail with an error



Suppose main function is forgotten

A screenshot of the Visual Studio Code interface. The terminal window shows the following output:

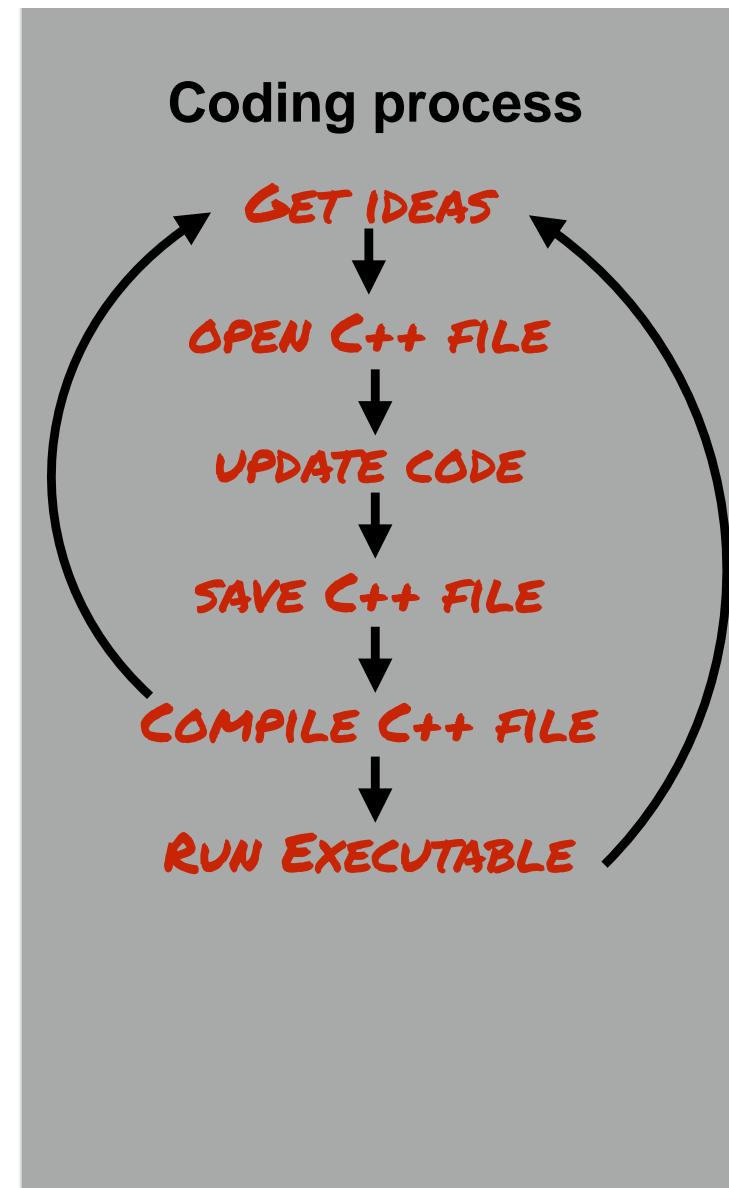
```
terminal  Help          hello.cpp - rob102-path-planning - Visual Studio Code
C:\Helloapp 1 X
C:\Users\logan\Documents\Directory for My Programs> C:\hello.cpp
1 #include <iostream>
2
3 int main()
4 {
5     std::cout << "Hello World" << "\n";
6     std::cout << "Chad is in Robotics 102";
7 }
8
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp --o hello
hello.cpp:3:1: error: expected unqualified-id before '{' token
  3 |
   |
PS C:\Users\logan\Documents\Directory for My Programs>
```

The code editor shows a file named `hello.cpp` with the following content:

```
#include <iostream>
int main()
{
    std::cout << "Hello World" << "\n";
    std::cout << "Chad is in Robotics 102";
}
```

A large orange rectangular overlay with the word "Oops" in white is centered over the code editor area.

Compilation will fail with an error

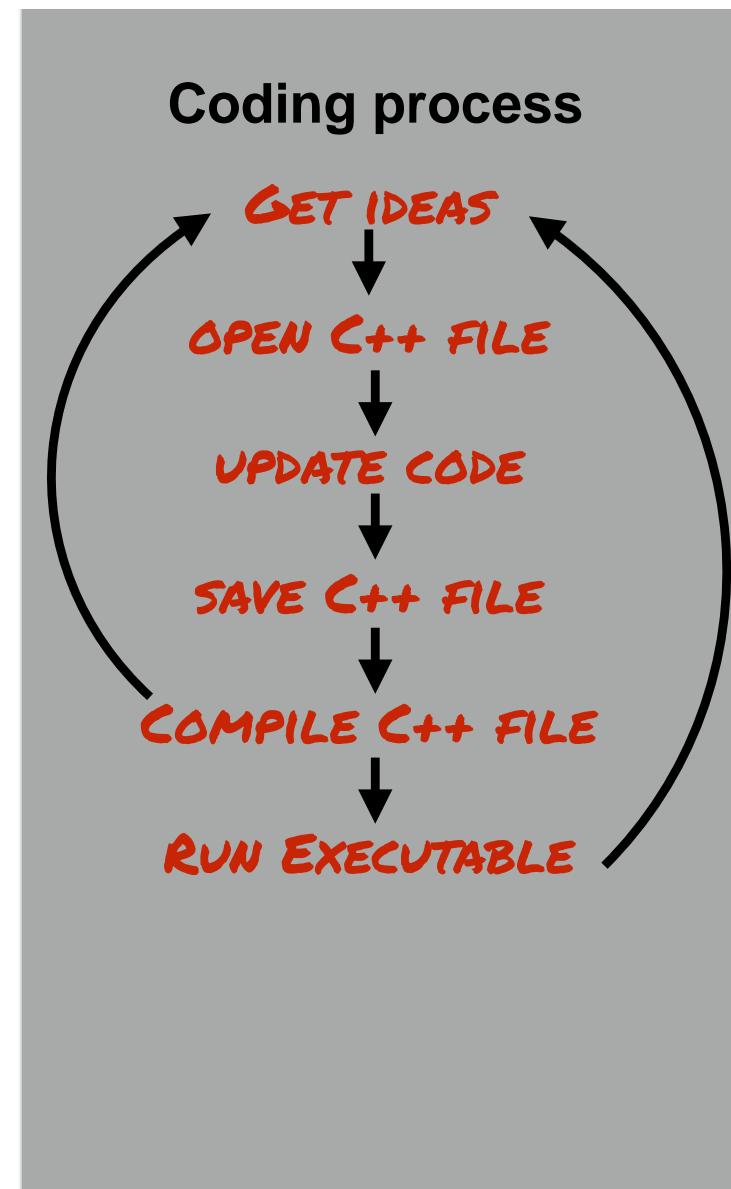


Suppose needed library is forgotten

A screenshot of Visual Studio Code showing a C++ file named `hello.cpp`. The code contains a simple `main` function that prints "Hello World" and "Chad is in Robotics 182". A red box highlights the first line of the code. The terminal below shows the compilation command `g++ hello.cpp -o hello` followed by several errors from the compiler. The errors indicate that `cout` is not a member of `std`, which is defined in the `<iostream>` header. The terminal also shows the inclusion of `<iostream>` in the code. The status bar at the bottom indicates the code has 6 lines and 2 columns.

```
C:\Users\logan> g++ hello.cpp -o hello
hello.cpp: In function 'int main()':
hello.cpp:4:10: error: 'cout' is not a member of 'std'
    4 |     std::cout << "Hello World" << "\n";
      |
hello.cpp:1:1: note: 'std::cout' is defined in header '<iostream>'; did you forget to '#include <iostream>?'
*** | #include <iostream>
   1 |
hello.cpp:5:10: error: 'cout' is not a member of 'std'
    5 |     std::cout << "Chad is in Robotics 182";
      |
hello.cpp:5:10: note: 'std::cout' is defined in header '<iostream>'; did you forget to '#include <iostream>?'
PS C:\Users\logan>
```

Compilation will fail with errors



Suppose I am just careless

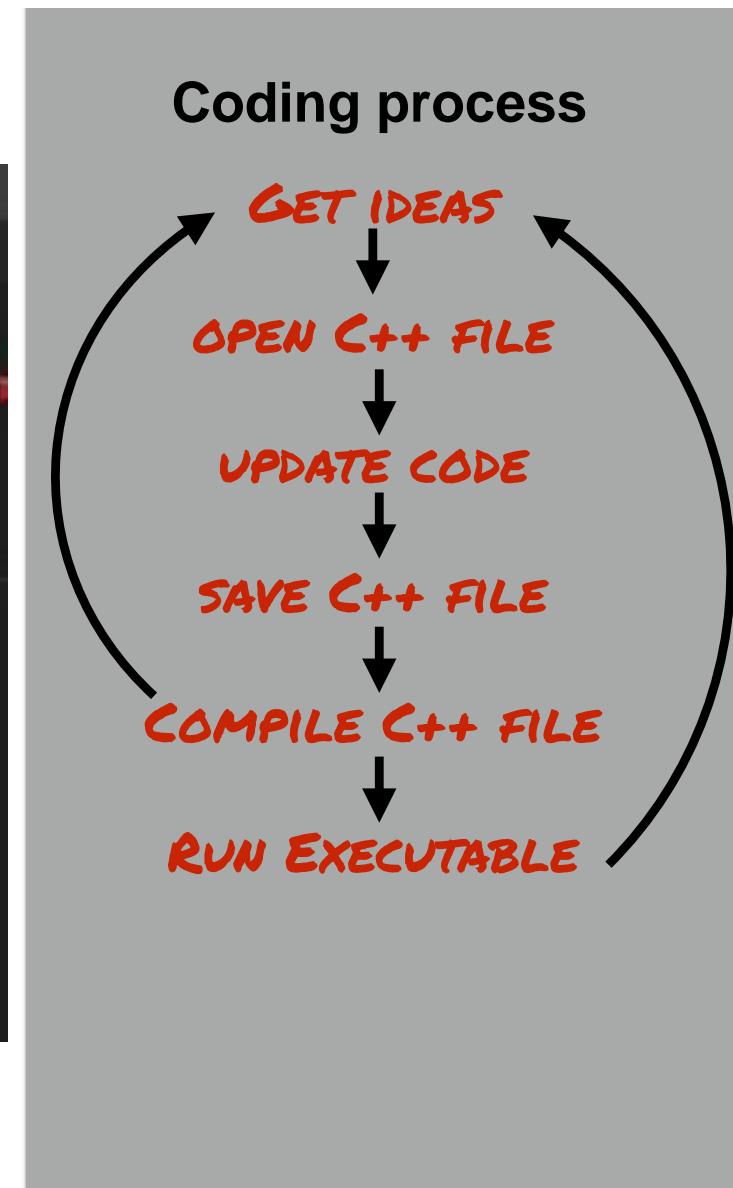
A screenshot of the Visual Studio Code interface. The title bar says "hello.cpp - rob102-path-planning - Visual Studio Code". The code editor shows a C++ file with the following code:

```
Terminal Help
hello.cpp - rob102-path-planning - Visual Studio Code
C:\Users\logan\Documents\Directory for My Programs> C:\hello.cpp > main()
1 #include <iostream>
2 int main()
3 {
4     whatever text because I feel like it
5     std::cout << "Hello World" << "\n";
6     std::cout << "Chad is in Robotics 182";
7 }
```

The word "whatever" is highlighted with a red box, indicating a syntax error. A large orange "Oops" button is overlaid on the center of the screen. The terminal below shows the command `g++ hello.cpp -o hello` and the resulting compilation error:

```
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
hello.cpp: In function 'int main()':
hello.cpp:4:5: error: 'whatever' was not declared in this scope
    4 |     whatever text because I feel like it
      |             ^
PS C:\Users\logan\Documents\Directory for My Programs> [ ]
```

Compilation will fail with an error



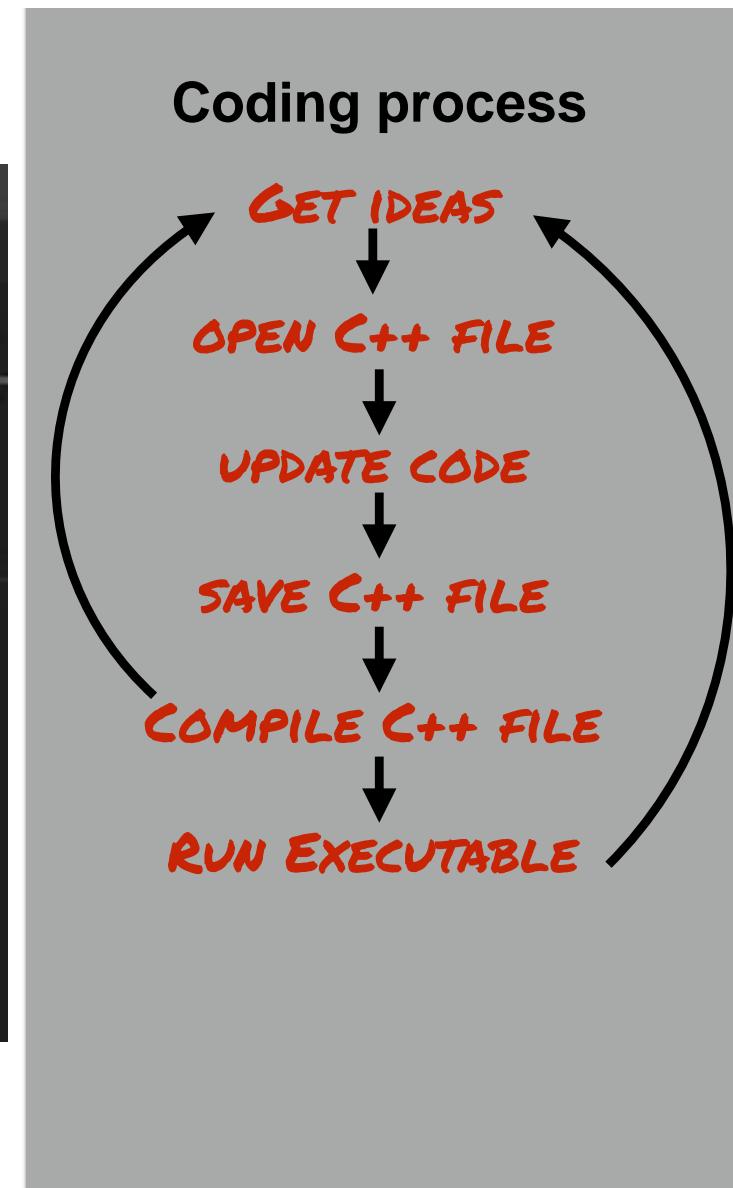
Suppose I am just careless

A screenshot of Visual Studio Code showing a C++ file named `hello.cpp`. The code contains a `cout` statement that prints two lines of text. A red box highlights the second line of the code, which is a comment: `// whatever text because I feel like it`. The terminal below shows the program being compiled and run, and it outputs both lines of text as expected.

```
Terminal Help          hello.cpp - rob102-path-planning - Visual Studio Code
C: hello.cpp X
C: > Users > logan > Documents > Directory for My Programs > C: hello.cpp > main()
1 #include <iostream>
2 int main()
3 {
4     // whatever text because I feel like it
5     std::cout << "Hello World" << "\n";
6     std::cout << "Chad is in Robotics 102";
7 }

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
powershell + ~ [ ] ^ X
PS: C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS: C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello World
Chad is in Robotics 102
PS: C:\Users\logan\Documents\Directory for My Programs>
```

Fixed



That can be “commented out”

C++ Comments

**Comments are ignored by the compiler
and not included in the program**

```
#include <iostream>
/*
    This is a multi-line comment. It is ignored by my program.
    This is our first program.
    The program code below prints messages to the screen.
*/

int main()
{
    std::cout << "Hello World" << "\n"; // A single-line comment
    std::cout << "Chad is in Robotics 102"; // "\n" creates a new line
}
```

hello.cpp

**Anything after double slashes on a line
is ignored as a comment**

Michigan Robotics 102 - robotics102.org

C++ Comments

**Comments are ignored by the compiler
and not included in the program**

Anything in between delimiters /* and */ is ignored as a comment

```
#include <iostream>
/*
    This is a multi-line comment. It is ignored by my program.
    This is our first program.
    The program code below prints messages to the screen.
*/
int main()
{
    std::cout << "Hello World" << "\n"; // A single-line comment
    std::cout << "Chad is in Robotics 102"; // "\n" creates a new line
}
```

hello.cpp

**Anything after double slashes on a line
is ignored as a comment**

Michigan Robotics 102 - robotics102.org

Your code is a battleground

```
#include <iostream>
/*
    This is a multi-line comment. It is ignored by my program.
    This is our first program.
    The program code below prints messages to the screen.
*/

int main()
{
    std::cout << "Hello World" << "\n"; // A single-line comment
    std::cout << "Chad is in Robotics 102"; // "\n" creates a new line
}
```

hello.cpp

Your code is a battleground

```
#include <iostream>
/* Hello World - A first C++ Program
   Copyright 2021 Odest Chadwicke Jenkins at the University of Michigan
   Licensed under Michigan Honor License in the LICENSE file and
   available to view at https://autorob.org/MichiganHonorLicense.txt
*/
int main()
{
    std::cout << "Hello World" << "\n"; // A single-line comment
    std::cout << "Chad is in Robotics 102"; // "\n" creates a new line
}
```

hello.cpp

```
#include <iostream>
/* Hello World - A first C++ Program
Copyright 2021 Odest Chadwicke Jenkins at the University of Michigan
Licensed under Michigan Honor License in the LICENSE file and
available to view at https://autorob.org/MichiganHonorLicense.txt
*/
int main()
{
    std::cout << "Hello World" << "\n"; // A single-line comment
    std::cout << "Chad is in Robotics 102"; // "\n" creates a new line
}
```

hello.cpp

LICENSE

The Michigan Honor License

This unvetted license below is called the "Michigan Honor License" as the 3-Clause BSD License plus two clauses for academic integrity.

Copyright <YEAR> <COPYRIGHT HOLDER>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

```
#include <iostream>
/* Hello World - A first C++ Program
Copyright 2021 Odest Chadwicke Jenkins at the University of Michigan
Licensed under Michigan Honor License in the LICENSE file and
available to view at https://autorob.org/MichiganHonorLicense.txt
*/
int main()
{
    std::cout << "Hello World" << "\n"; // A single-line comment
    std::cout << "Chad is in Robotics 102"; // "\n" creates a new line
}
```

hello.cpp

LICENSE

The Michigan Honor License

This unvetted license below is called the "Michigan Honor License" as the 3-Clause BSD License plus two clauses for academic integrity.

Copyright <YEAR> <COPYRIGHT HOLDER>

This year and your name

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

LICENSE

The Michigan Honor License

This unvetted license below is called the "Michigan Honor License" as the 3-Clause BSD License plus two clauses for academic integrity.

BSD 3-Clause License

<https://opensource.org/licenses/BSD-3-Clause>

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
- Manuscripts and publications using this source code or its binary forms must properly cite the this project and its author(s).
- Redistributions of source code, if used for credit in an academic program, must retain the academic integrity honor pledge below.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE

LICENSE

The Michigan Honor License

This unvetted license below is called the "Michigan Honor License" as the 3-Clause BSD License plus two clauses for academic integrity.

Copyright <YEAR> <COPYRIGHT HOLDER>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Michigan Engineering Honor Code

[**https://bulletin.engin.umich.edu/rules/**](https://bulletin.engin.umich.edu/rules/)

4. Redistributions of source code, if used for credit in an academic program, must retain the following honor pledge, and only append the names all individuals who contributed modifications.

the academic integrity honor pledge below.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE

LICENSE

The Michigan Honor License

This unvetted license below is called the "Michigan Honor License" as the 3-Clause BSD License plus two clauses for academic integrity.

Copyright <YEAR> <COPYRIGHT HOLDER>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

Attribution similar to CC BY 4.0 License

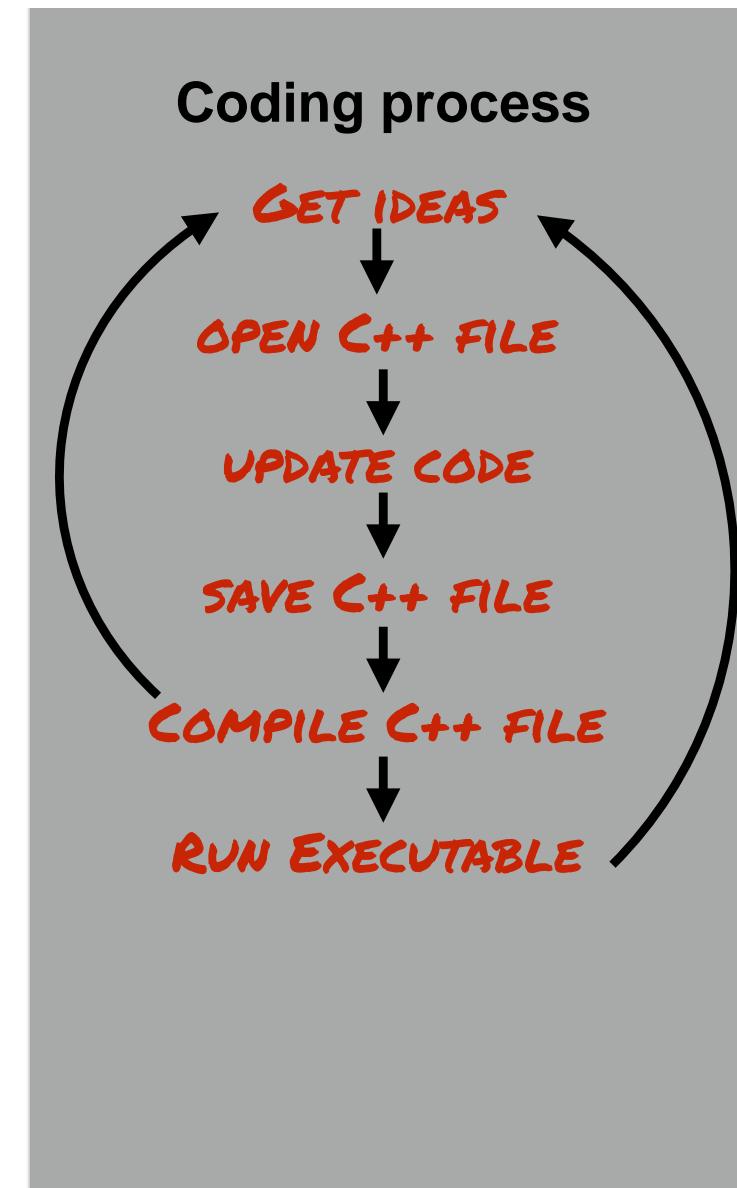
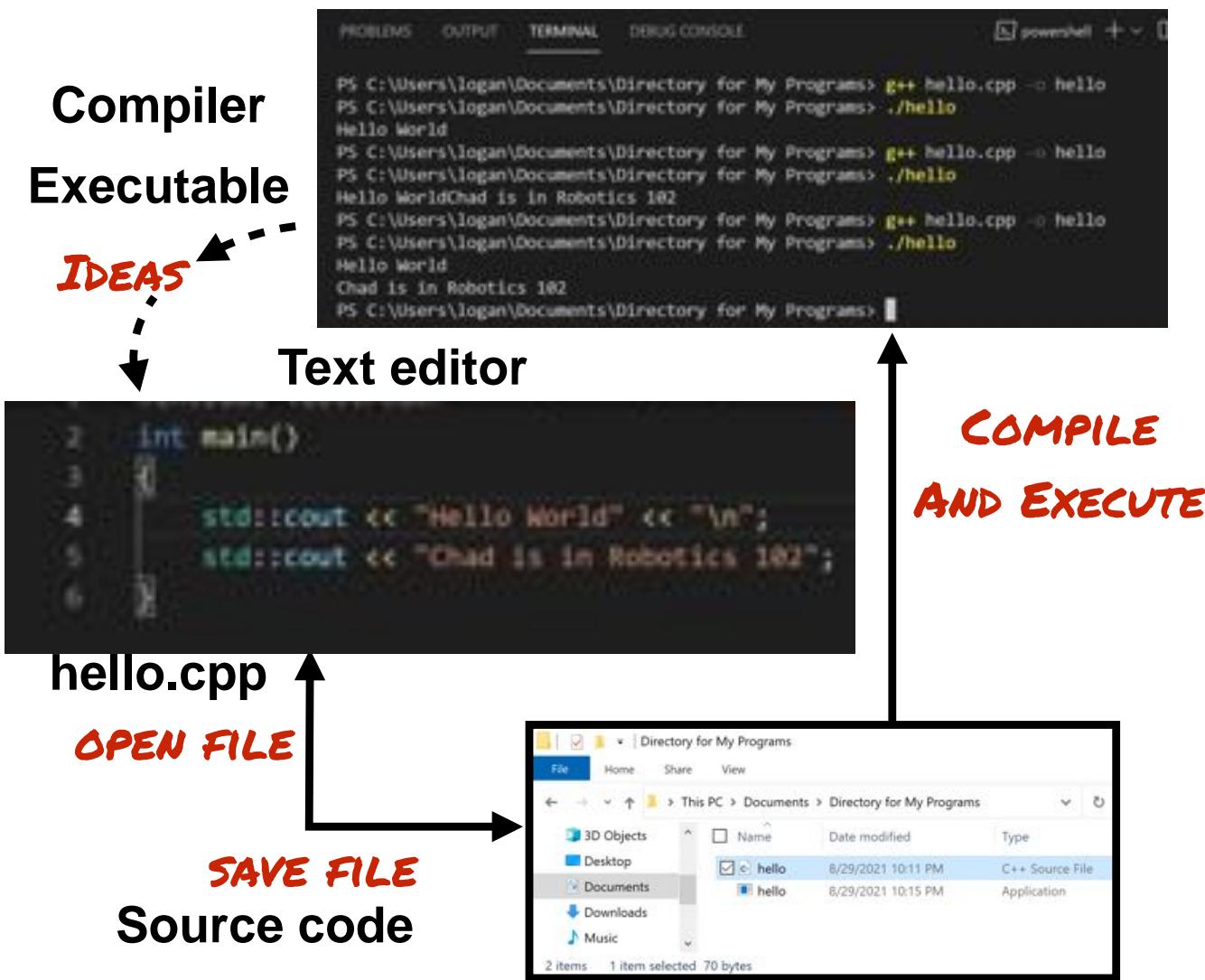
<https://creativecommons.org/licenses/by/4.0/>

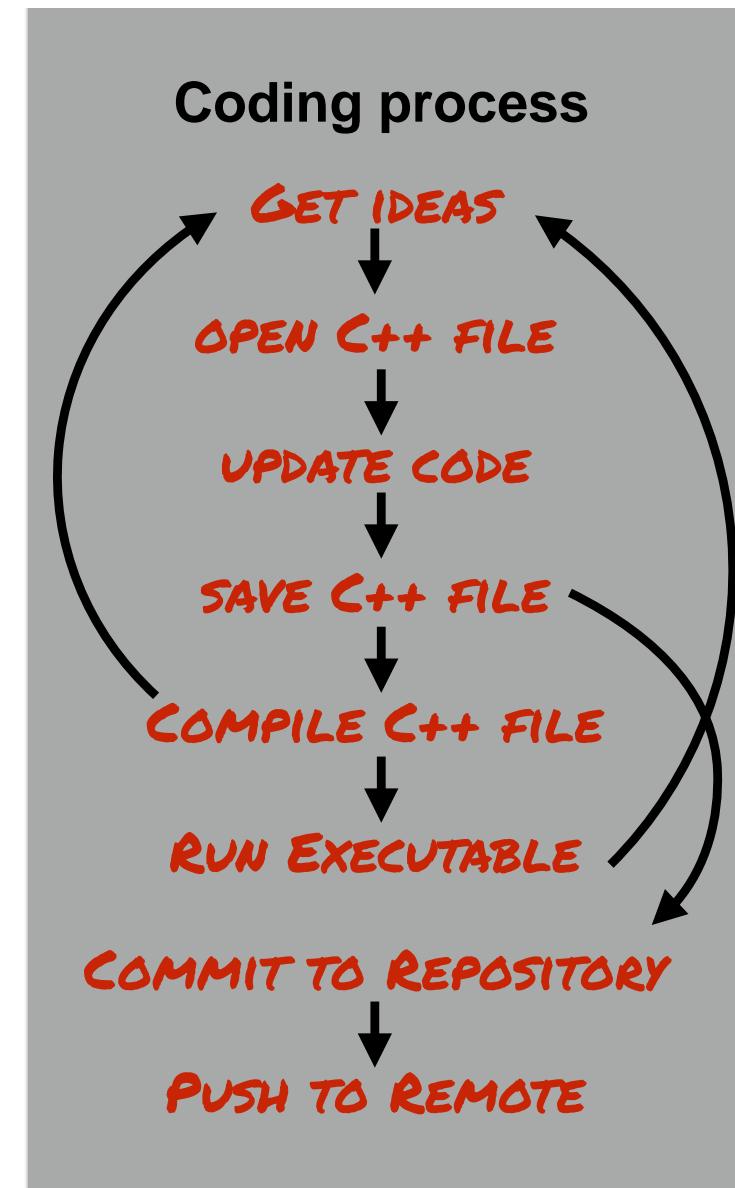
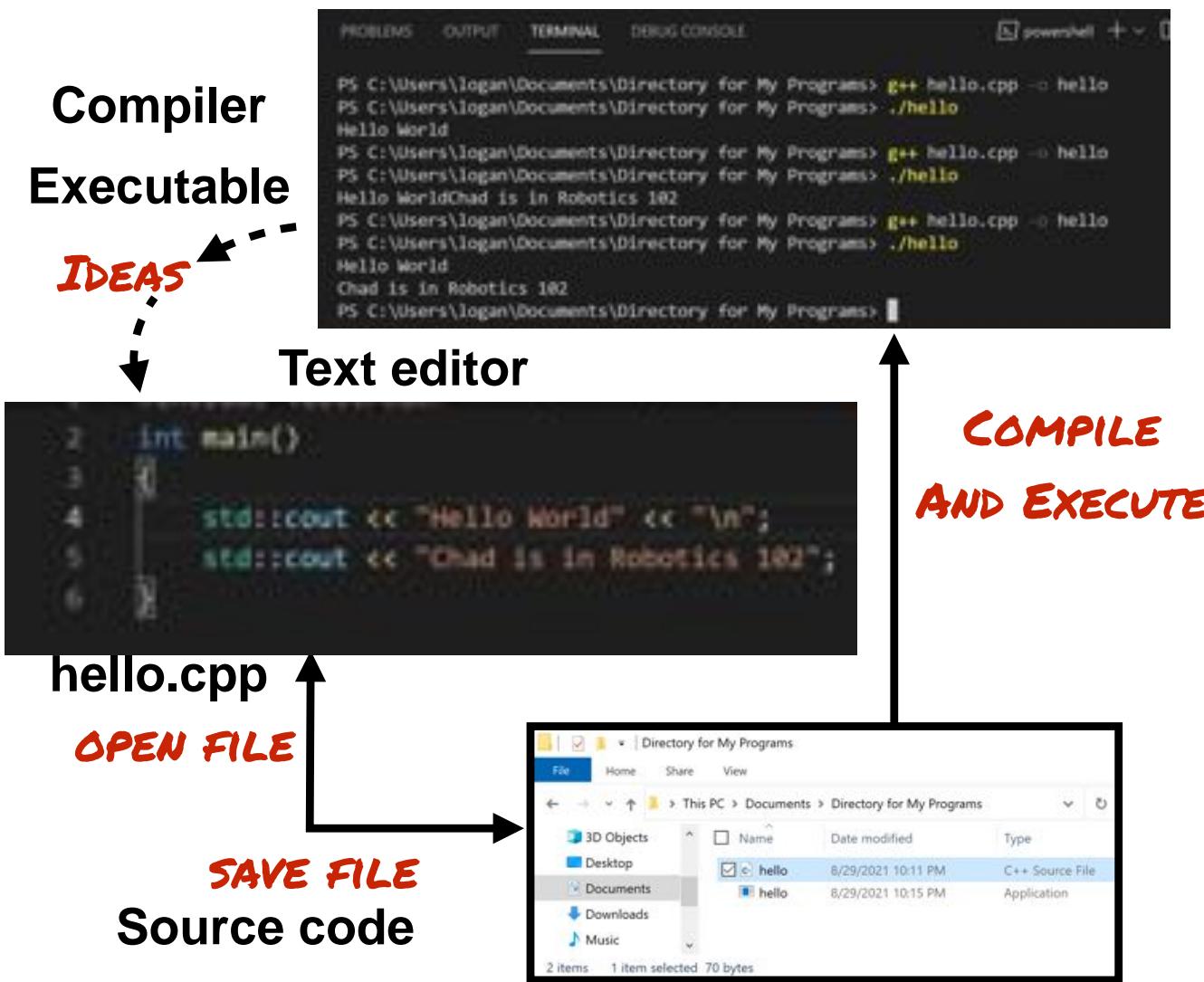
5. Manuscripts and publications using this source code or its binary forms must properly cite this project and its author(s).

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE

Share your code with the world

and your future self





Compiler
Executable
IDEAS

Text editor

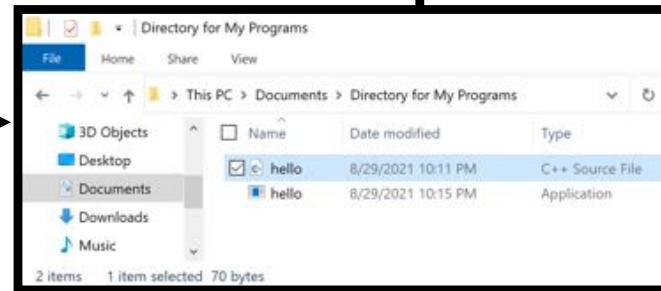
```
2 int main()
3 {
4     std::cout << "Hello World" << "\n";
5     std::cout << "Chad is in Robotics 102";
6 }
```

hello.cpp

OPEN FILE

SAVE FILE

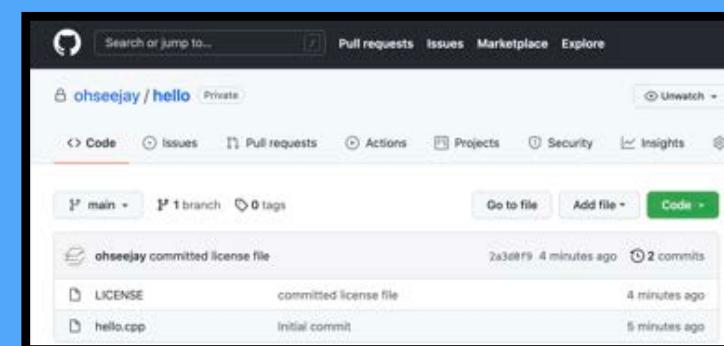
Source code



**COMPILE
AND EXECUTE**

```
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello World
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello WorldChad is in Robotics 102
PS C:\Users\logan\Documents\Directory for My Programs> g++ hello.cpp -o hello
PS C:\Users\logan\Documents\Directory for My Programs> ./hello
Hello World
Chad is in Robotics 102
PS C:\Users\logan\Documents\Directory for My Programs>
```

git repository
store history of
code changes
and collaborate with others



PUSH REPO

PULL REPO

Version Control Using git

What is Version Control?

- Maintains a past history of changes for your code (or any project)
- History of changes (or “commits”) maintained in a repository
- Basic workflow
 - Code is “checked out” (or “pulled”) from a repository, then modified
 - These updates are then “checked in” (or “committed”) to the repository
 - Repository maintains history as “diffs”, the changes between before and after checking in a commit

For example... ocj's TED talk

https://www.ted.com/talks/henry_evans_and_chad_jenkins_meet_the_robots_for_humanity?language=en

TED Watch Discover Attend Participate About Search... Log in

Henry Evans and Chad Jenkins:

Meet the robots for humanity

TEDxMidAtlantic · 10:21 · Filmed Oct 2013

28 subtitle languages ⓘ View interactive transcript

A large play button icon is overlaid on the video player. Below the video player, there's a small image of a quadcopter drone flying. To the right of the video player, a man in a brown shirt and blue jeans is standing on a stage, gesturing with his hands as if speaking. In the background, a large red "TEDx" logo is visible.

Watch later Favorite Play Download Rate

Share this idea

[Facebook](#) [Twitter](#) [Email](#) [Embed](#) [More](#)

1,145,408 Total views

Share this talk and track your influence!

TED Talks are free thanks to support from **Infosys**

https://www.ted.com/talks/henry_evans_and_chad_jenkins_meet_the_robots_for_humanity?language=en

https://www.ted.com/talks/henry_evans_and_chad_jenkins_meet_the_robots_for_humanity?language=en

TED Watch Discover Attend Participate About Search... Log in

Henry Evans and Chad Jenkins:

Meet the robots for humanity

TEDxMidAtlantic · 10:21 · Filmed Oct 2013

28 subtitle languages ⓘ View interactive transcript

A large play button icon is overlaid on the video frame. In the bottom right corner of the video frame, there is a white rectangular box containing the handwritten text "OCJ IN WASHINGTON DC". To the right of the video frame, there is a vertical column of four circular icons with accompanying text: "Watch later", "Favorite", "Play", and "Download".

Share this idea

[Facebook](#) [Twitter](#) [Email](#) [Embed](#) [More](#)

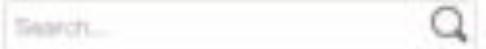
1,145,408 Total views

Share this talk and track your influence!

TED Talks are free thanks to support from **Infosys**

https://www.ted.com/talks/henry_evans_and_chad_jenkins_meet_the_robots_for_humanity?language=en

https://www.ted.com/talks/henry_evans_and_chad_jenkins_meet_the_robots_for_humanity?language=en

TED Watch Discover Attend  Log in

HENRY EVANS IN PALO ALTO CA 

Meet the robots for humanity

TEDxMidAtlantic · 10:21 · Filmed Oct 2013

28 subtitle languages  View interactive transcript 

OCJ IN WASHINGTON DC 

 Watch later  Favorite  Download 

Share this idea



1,145,408 Total views

Share this talk and track your influence!

TED Talks are free thanks to support from

Infosys



View history of changes

The screenshot shows a GitHub repository page for 'tutorial_rosbridge_ardrone'. At the top, there's a navigation bar with links for 'Pull requests', 'Issues', and 'Gist'. Below the navigation bar, the repository name 'odestcj / tutorial_rosbridge_ardrone' is displayed, along with options to 'Unwatch', 'Star', and 'Fork'. A red box highlights the '7 commits' link. The main content area shows a list of commits:

Commit	Message	Date
alicef	Create Fly ardrone through ROS Hydro	May 15, 2014
rosbridge_ardrone	Initial commit	3 years ago
rosbridge_ardrone_buttons	added actual drone_browser_teleop.html with buttons	3 years ago
Fly ardrone through ROS H...	Create Fly ardrone through ROS Hydro	2 years ago
README.md	Added rosbridge tutorial for ROS Fuerte	2 years ago
rosbridge_ardrone.launch	added launch file	3 years ago
README.md		

Below the commit list, the text 'tutorial_rosbridge_ardrone' is displayed.

https://github.com/odesto/tutorial_rosbridge_ardrone

odesto / tutorial_rosbridge_ardrone

Code Issues Pull requests Wiki Pulse Graphs Settings

Branch: master

Commits on May 15, 2014

 Create Fly ar.drone through ROS Hydro [diff](#)
alicef committed on May 15, 2014

Commits on May 6, 2014

 Added rosbridge tutorial for ROS Fuerte
odesto committed on May 6, 2014

 Create README.md
odesto committed on May 6, 2014

Commits on Apr 30, 2013

 added launch file
odesto committed on Apr 30, 2013

Commits on Apr 18, 2013

 added actual drone_browser_teleop.html with buttons
odesto committed on Apr 18, 2013

 Fixed mappings of buttons and keys to drone commands
odesto committed on Apr 18, 2013

Commits on Apr 5, 2013

View history of changes

Large open source projects...



3D INTERACTIONS

USING THE LATEST IN WEBGL



MULTI-PLATFORM SUPPORT

HARNESSING THE POWER OF ROS



TOWARDS COMPATIBILITY

MORE BROWSERS, MORE ROBOTS.

ROBOT WEB ARCHITECTURE

BRIDGING ROBOTS AND THE WEB

ROSBRIDGE AS A TRANSPORT

USING JSON TO SPEAK TO YOUR ROBOT

A variety of routes are available for architecting a robot web

While ROS works great for applications on the robot, another layer is

GitHub, Inc. 6/8 https://github.com/RobotWebTools/rosbridge_suite

Search

This repository

Pull requests Issues Gist

Unwatch 26 Unstar 42 Fork 71

Code Issues 16 Pull requests 2 Pulse Graphs Settings

Server implementations of the rosbridge v2 Protocol <http://robotwebtools.org/> — Edit

523 commits 7 branches 37 releases 33 contributors

623 commits 33 contributors

New file Find file SSH git@github.com:RobotWebTools/rosbridge_suite.git

set/UDP ...

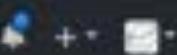
File	Description	Time Ago
rosapi	Update proxy.py	4 months ago
rosbridge_library	0.7.13	6 months ago
rosbridge_server	enable udp	2 months ago
rosbridge_suite	0.7.13	6 months ago
.gitignore	cleanup of old misc. files from old merge of new features	2 years ago
.travis.yml	ci: test with and without ujson	a year ago
AUTHORS.md	authors and license added	2 years ago
CHANGELOG.md	update the change log	2 years ago
LICENSE	authors and license added	2 years ago
README.md	Update README.md	a year ago
ROSBRIDGE_PROTOCOL...	protocol documented for advertise service functions	a year ago

September 2016



This repository / Issues

Pull requests Issues Marketplace Explore



RobotWebTools / rosbridge_suite

Watch

38

Star

106

Fork

114

Code

Issues 36

Pull requests 1

Projects 0

Insights

Server implementations of the rosbridge v2 Protocol <http://robotwebtools.org/>

September 2017

623 commits

9 branches

48 releases

47 contributors

BSD-3-Clause



623 commits

47 contributors

rosapi	0.8.3	7 hours ago
rosbridge_library	0.8.3	7 hours ago
rosbridge_server	0.8.3	7 hours ago
rosbridge_suite	0.8.3	7 hours ago
.gitignore	Gitignore vim swapfile	a year ago
.travis.yml	Cleaning up travis configuration (#283)	2 months ago
AUTHORS.md	authors and license added	3 years ago
CHANGELOG.md	update the change log	4 years ago
LICENSE	authors and license added	3 years ago
README.md	Update README.md	3 years ago

GitHub, Inc. (US) https://github.com/RobotWebTools/rosbridge_suite ... Search

RobotWebTools / rosbridge_suite

Code Issues Pull requests Insights Settings

Server Implementations of the rosbridge v2 Protocol <http://robotwebtools.org/>

Add topics

653 commits 12 branches 52 releases All 58 contributors

653 commits

All 58 contributors

Author	Commit Message	Date
rosapi	Fix a few problems (#350)	25 days ago
rosbridge_library	use package format 2, remove unnecessary dependencies (#348)	25 days ago
rosbridge_server	Fix a few problems (#350)	25 days ago
rosbridge_suite	use package format 2, remove unnecessary dependencies (#348)	25 days ago
.gitignore	Gitignore vim swapfile	2 years ago
.travis.yml	Fix Travis config (#311)	8 months ago
AUTHORS.md	authors and license added	5 years ago
CHANGELOG.md	update the change log	5 years ago

102.org

September 2018

RobotWebTools / rosbridge_suite

Unwatch

46

Unstar

251

Fork

221

Code

Issues 47

Pull requests 8

Security

Insights

Settings

Server Implementations of the rosbridge v2 Protocol <http://robotwebtools.org/>

Edit

Manage topics

685 commits

12 branches

69 releases

All 66 contributors

View license



685 commits

ghome 0.11.3 release (#424) ...

All 66 contributors

Clone or download

Latest commit dates 28 days ago

.github

Add GitHub issue template and TROUBLESHOOTING.md (x397)

5 months ago

.rosapi

0.11.3 release (#424)

28 days ago

rosbridge_library

0.11.3 release (#424)

28 days ago

rosbridge_msgs

0.11.3 release (#424)

28 days ago

rosbridge_server

0.11.3 release (#424)

28 days ago

rosbridge_suite

0.11.3 release (#424)

28 days ago

.gitignore

Gitignore vim swapfile

3 years ago

.travis.yml

Travis CI: Look for Python syntax errors and undefined name (#420)

2 months ago

AUTHORS.md

authors and license added

6 years ago

September 2019

August 2020

RobotWebTools / rosbridge_suite

Code Issues (78) Pull requests Actions Security Insights Settings

develop 18 branches 68 tags Go to file Add file + ± Code + About

722 commits

hynneva add ros 1 github actions (#525) ✓ 6 days ago

.github add ros 1 github actions (#525)

.rosapi Fixed filter_glob for noetic (#506)

rosbridge_library possible fix for error when working with RosShares, TypeError: can only concat str to str (#510) 2 months ago

rosbridge_msgs 0.11.9 3 months ago

rosbridge_server Error initialization with tornado. (#510) 3 months ago

rosbridge_suite 0.11.9 3 months ago

.gitignore .gitignore vim swapfile 4 years ago

.travis.yml noetic tests (#503) 3 months ago

AUTHORS.md Add myself to contributors 6 months ago

CHANGELOG.md update the change log 7 years ago

Dockerfile noetic tests (#503) 3 months ago

LICENSE authors and license added 7 years ago

View license

Releases 68 tags Create a new release

Packages No packages published Publish your first package

Contributors (8)

https://github.com/RobotWebTools/rosbridge_suite

Search or jump to... Pull requests Issues Marketplace Explore

RobotWebTools / rosbridge_suite

Issues 595 Pull requests 36 Discussions Actions Security Insights Settings

August 2021

1 7 commits 78 branches 78 tags Go to file Add file

jhawkes 1.0.8 ✓ e37244a 4 days ago

miscellaneous files: fixed --select=E402,E731,E041,F821 (#621) 10 days ago

.github 1.0.8 4 days ago

rosapi 1.0.8 4 days ago

rosbridge_library 1.0.8 4 days ago

rosbridge_msgs 1.0.8 4 days ago

rosbridge_server 1.0.8 4 days ago

rosbridge_suite 1.0.8 4 days ago

.gitignore Add gzip raw compression support (#574) 27 days ago

AUTHORS.md fix: remove json encoding before setting string params (#527) 12 months ago

CHANGELOG.md Fix types discovered by codespell (#600) 12 days ago

LICENSE authors and license added 8 years ago

README.md Remove .travis.yml (#595) 12 days ago

roboteethools.org Readme View license

Releases 78 tags Create a new release

Contributors 35

https://github.com/RobotWebTools/rosbridge_suite

This repository Search

Pull requests Issues Gist

RobotWebTools / rosbridge_suite

Unwatch 28 Unstar 42 Fork 71

Code Issues 65 Pull requests 2 Pulse Graphs Settings

Branch: develop

Commits on Nov 12, 2015

Merge pull request #197 from xuhao1/UDP
xuhao1 committed on Nov 12, 2015

Commits on Nov 11, 2015

enable udp
xuhao1 committed on Nov 11, 2015

Commits on Nov 10, 2015

?

xuhao1 committed on Nov 10, 2015

Commits on Nov 9, 2015

Adding UDP
xuhao1 committed on Nov 9, 2015

Commits on Sep 28, 2015

Merge pull request #195 from rcoodddow/patch-1
rcoodddow committed on Sep 28, 2015

Commit History

CAN COPY OR REVERT TO ANY PAST STATE OF THE REPOSITORY

ics102.org

changelog updated

by develop at 0.7.13

robotis committed on Aug 14, 2015

1 parent 3cfc76b commit a2b3f869a67ema8e4d17a358d5346b464731544a

Showing 4 changed files with 36 additions and 0 deletions.

Unified Split

3 rosapi/CHANGELOG.rst

06 -23, 8 +23,11 @@ Changelog for package rosapi

23 23 0.7.0 (2014-12-02)

24 24 -----

25 25 + 0.7.13 (2015-08-14)

26 +-----

27 +** Fix catkin_lint issues

28 +** Contributors: Matt Vollrath

29 +*

30 31 0.7.12 (2015-04-07)

32 -----

33

14 rosbridge_library/CHANGELOG.rst

06 -34,8 +34,28 @@ Changelog for package rosbridge_library

34 34 * request_id --> id

35 35 * Contributors: Russell Toris

36 36 -----

37 37 +0.7.13 (2015-08-14)

38 +-----

39 +** Nevermind a_0

40 +** Add test_depend too (just in case)

41 +** Add dependency on python bson

42 +** Get parameter at encode time

Change log

**YOU CAN VIEW ALL CHANGES
MADE BETWEEN CONSECUTIVE COMMITS**

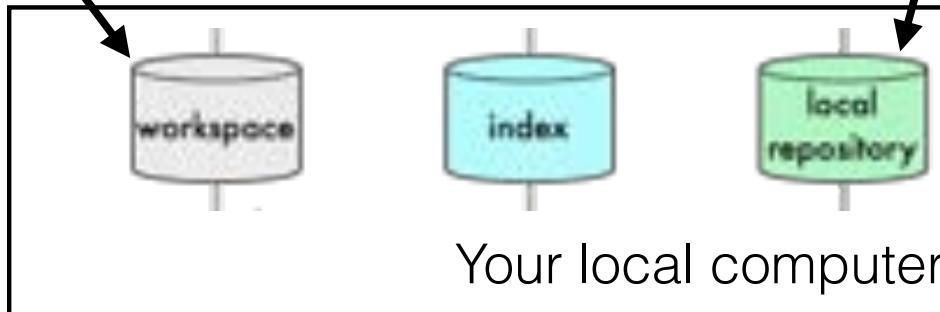
How does git work?

Git Data Transport Commands

<http://cstealle.com>

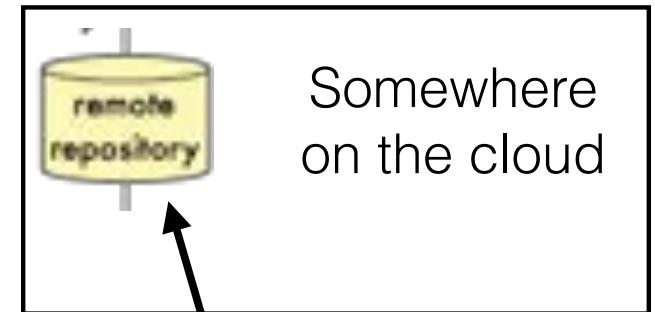
The directory where
you are working

(~/uname/reponame or
C:\Users\uname\reponame)



The repository on your
local computer

(~/uname/reponame/.git)



The repository on
a remote server

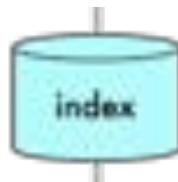
(http://github.com/username)

Git Data Transport Commands

<http://cstealle.com>

The directory where
you are working

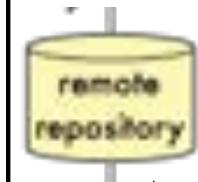
(~/uname/reponame or
C:\Users\uname\reponame)



Your local computer

The repository on your
local computer

(~/uname/reponame/.git)



Somewhere
on the cloud

This PC > Documents > Directory for My Programs		
Name	Date modified	Type
hello	8/30/2021 12:20 AM	C++ Source File
hello	8/30/2021 12:20 AM	Application
LICENSE	8/30/2021 9:53 PM	Text Document

The repository on
a remote server

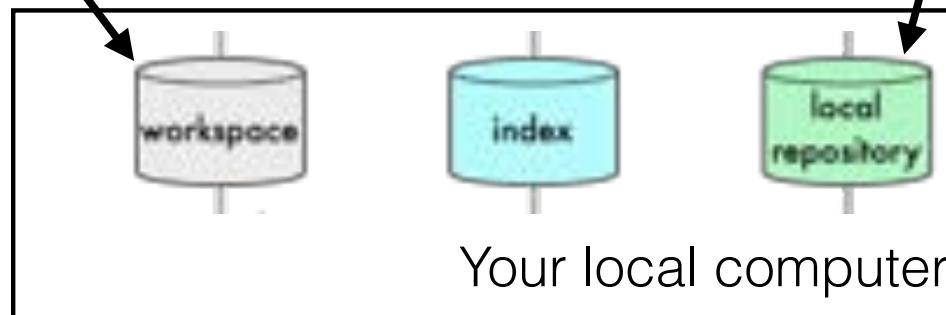
(<http://github.com/username>)

Git Data Transport Commands

<http://cstealle.com>

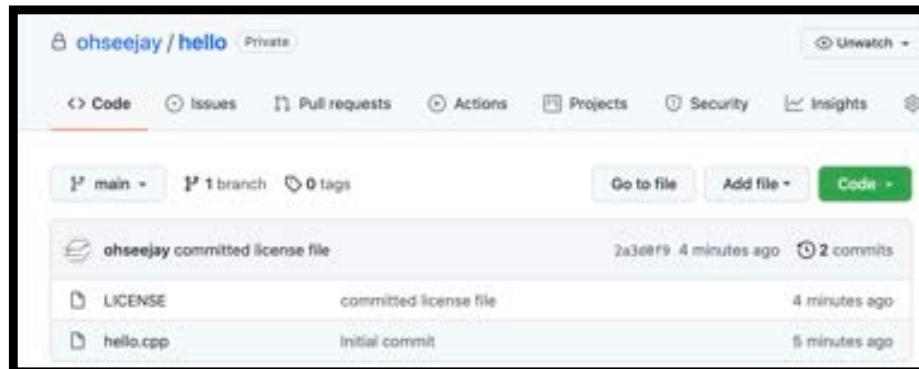
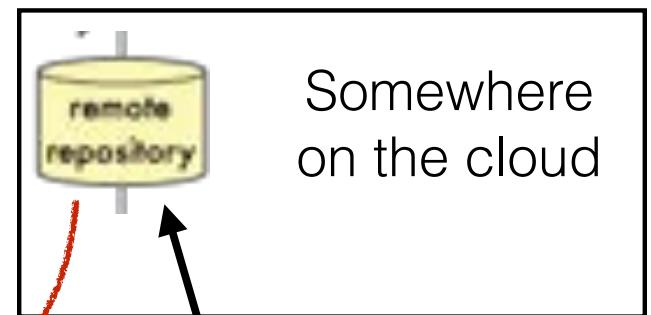
The directory where
you are working

(~/uname/reponame or
C:\Users\uname\reponame)



The repository on your
local computer

(~/uname/reponame/.git)

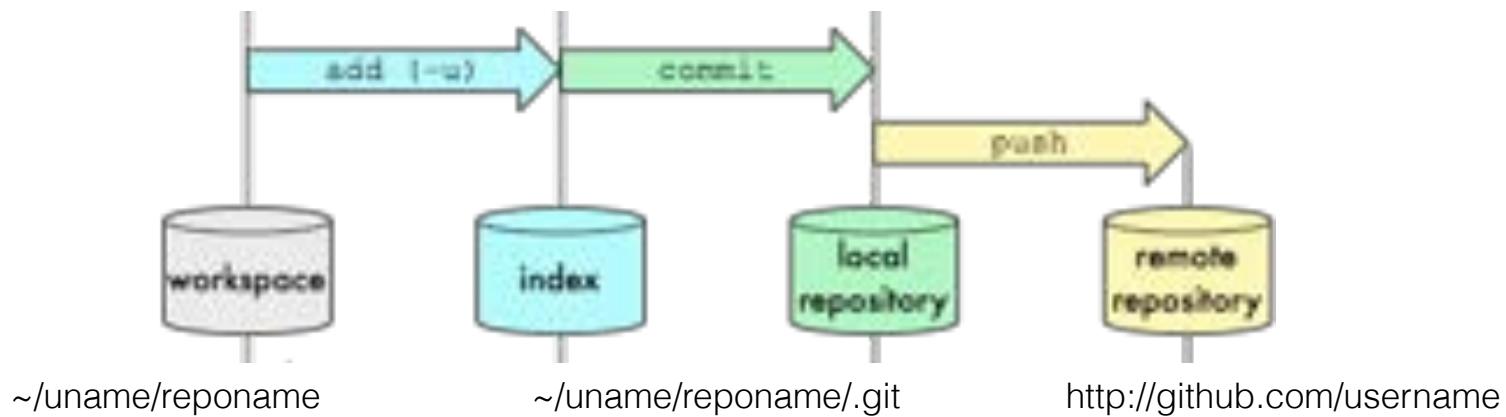


The repository on
a remote server

(<http://github.com/username>)

Git Data Transport Commands

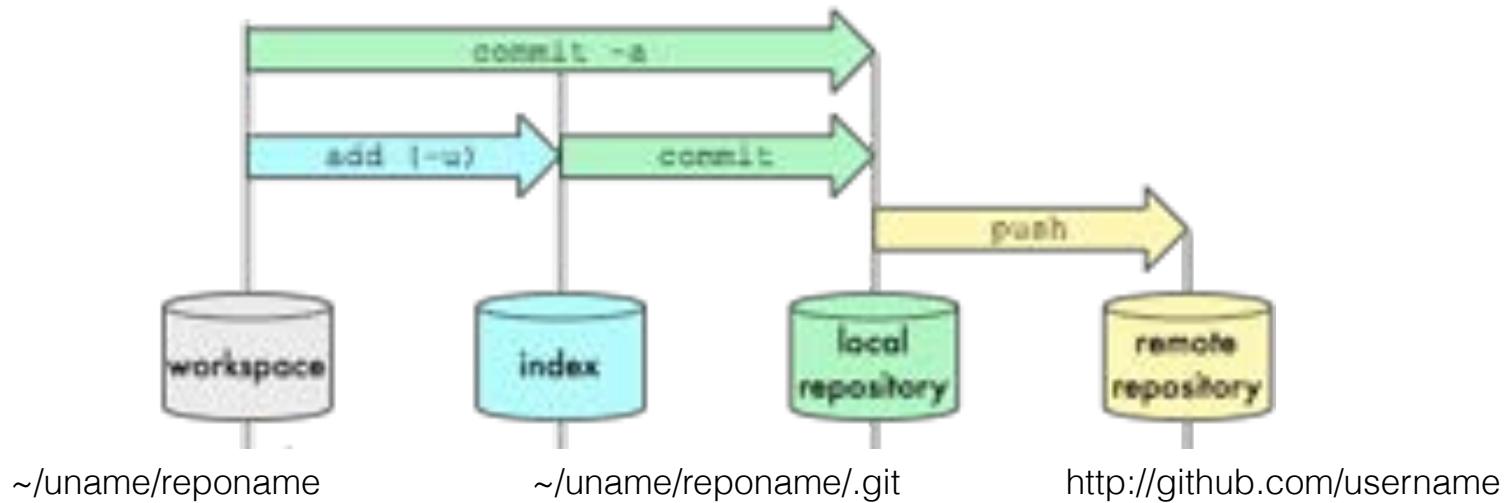
<http://cstealle.com>



After making local changes, you can add, commit, and push to your remote repository

Git Data Transport Commands

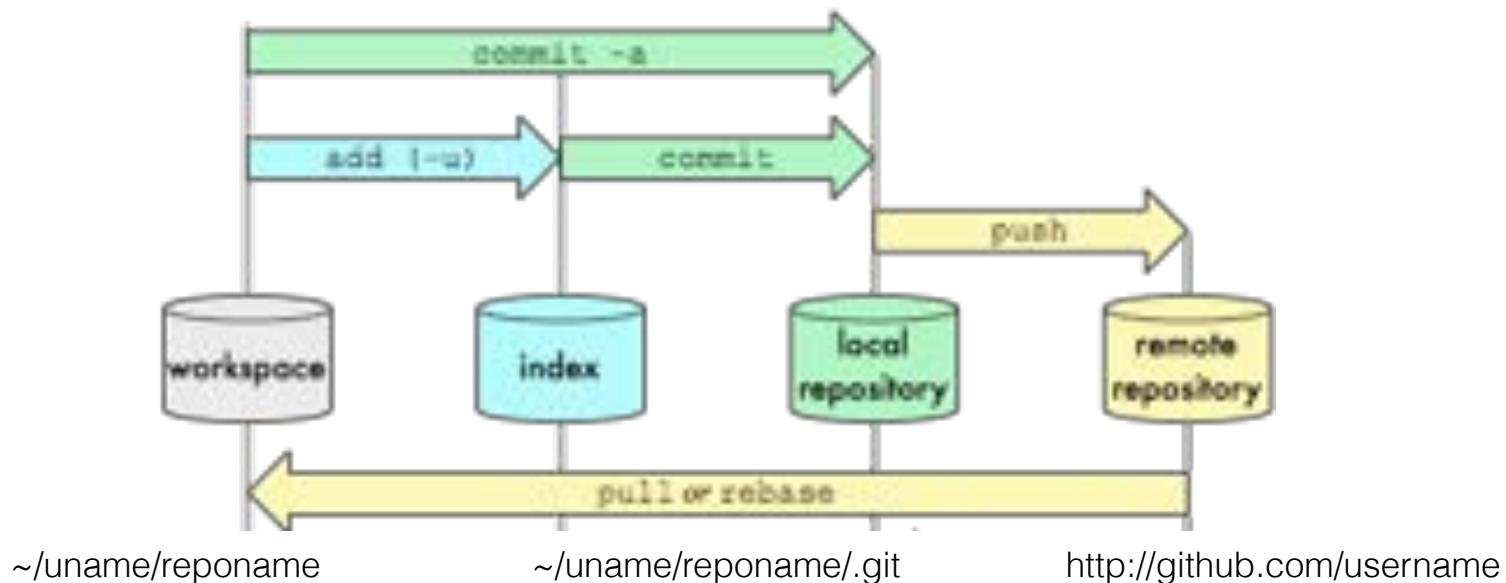
<http://cstealle.com>



If there are no files to add, just commit and push

Git Data Transport Commands

<http://cstealle.com>



A pull command updates the local workspace with changes from the remote repository

git basics: commands

- Push completed project to repository (or just to update)
 - add files to a repository: `git add <file listing>`
 - commit changes to local repo: `git commit -a -m "<msg>"`
 - push local changes to a remote repository: `git push`
- Pull to updates your local repository (and workspace) from remote
 - pull remote changes to a local repository: `git pull`

**Our first challenge:
Project 0**

Our first challenge: Project 0

Pocket Calculator



Our first challenge: Project 0

Pocket Calculator

```
$ ./calculator
Please type a number and press enter: 100
Please type a math operator (one of: + - * /): +
Please type a number and press enter: 2
100+2= 102
```

Actually,
it will look more like this

Next Lecture: Operators and Variables

```
#include <iostream>

/*
    Let's write a calculator program
*/

int main()
{
    std::cout << "What is 100 plus 2?" << "\n";
}
```

calculator.cpp (Version 00)