# INTRODUCTION TO C++

Dr. Alex Hill

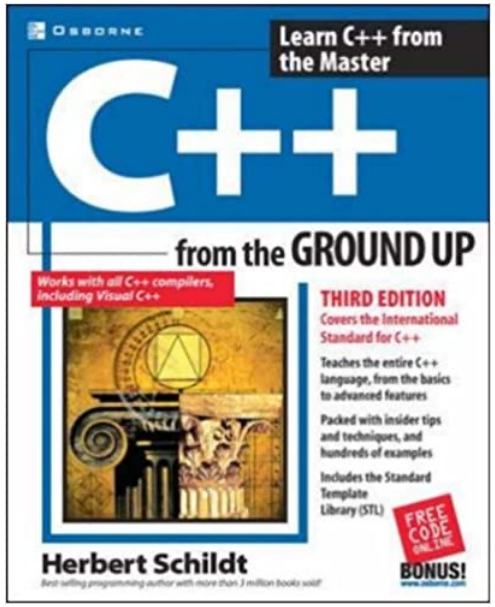
a.d.hill@liverpool.ac.uk

October 2023

# COURSE AIMS

- Introduce you to the C++ programming language
- Run through the syntax and basic operations
- Work through some examples together, increase fluency
- Introduction to Monte Carlo methods
- Complete a collaborative project using C++ with a Monte Carlo context

# RESOURCES



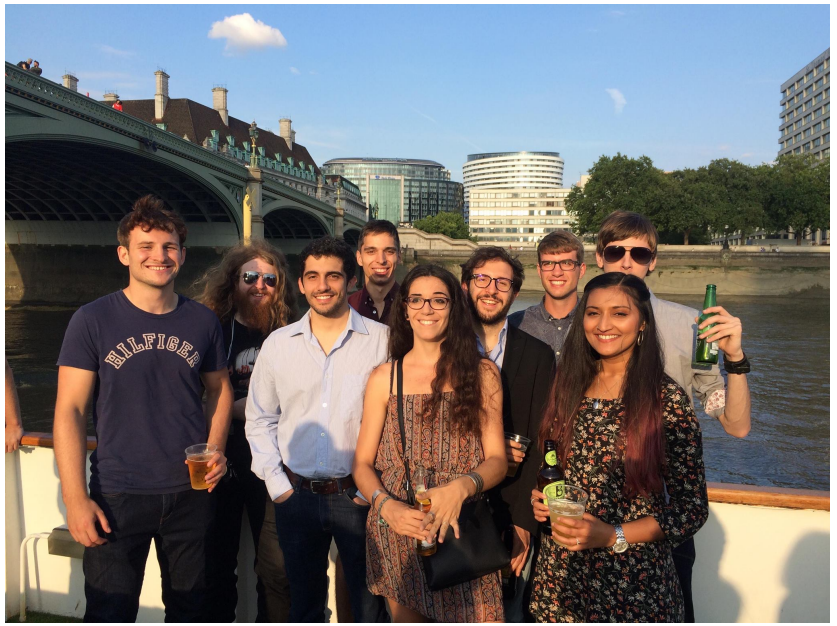

- https://alex-hill94.github.io/
- C++ from the ground up, Herbert Schildt
- Online compiler: https://www.programiz.com/cpp-programming/online-compiler/
- Online tutorials: https://www.w3schools.com/cpp
- Interview with C++ creator - https://www.youtube.com/watch?v=uTxRF5ag27A

# AIM OF WORKSHOP ONE

- Introductions
- History and philosophy of C++
- Get set up with a text editor and compiler
- Run a script

# INTRODUCTIONS

- Started PhD in astrophysics in 2017 at the ARI
- Part of the first LIV.DAT cohort (precursor to LIV.INNO)
- Researched cosmological simulations, now working on medical physics
- Primarily use Python

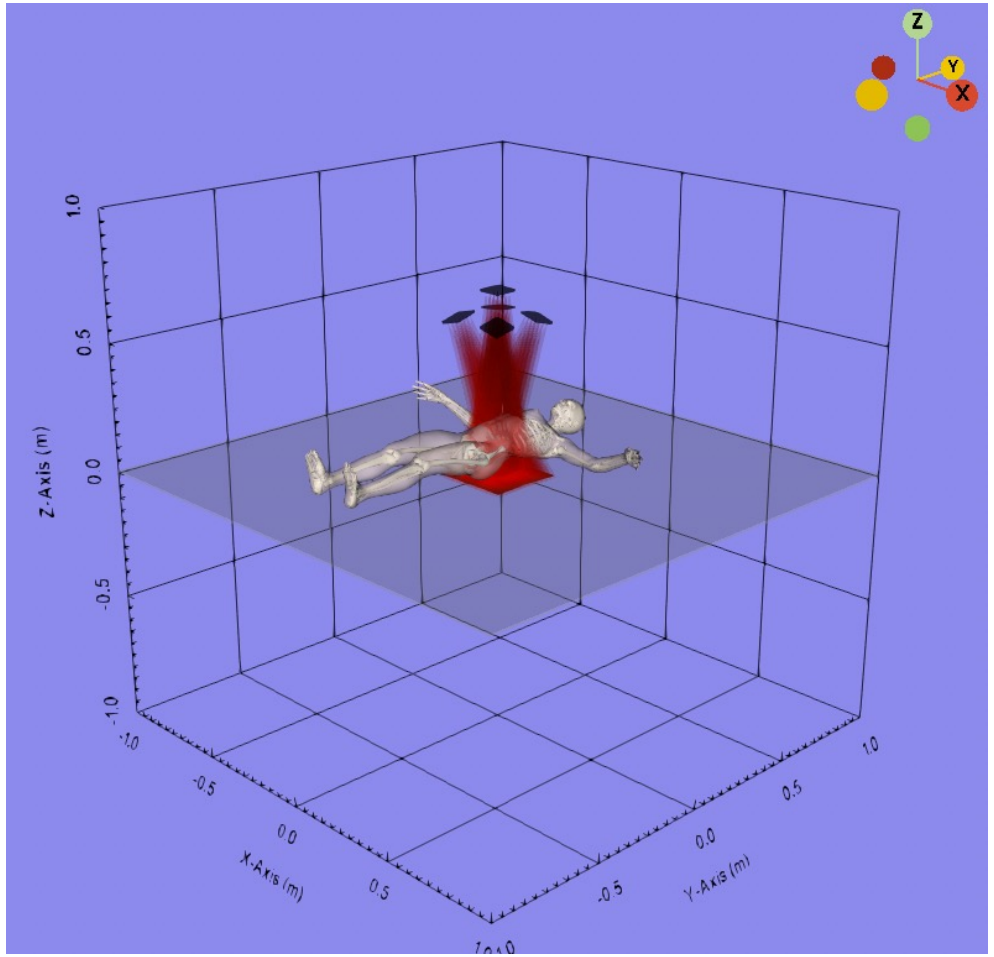


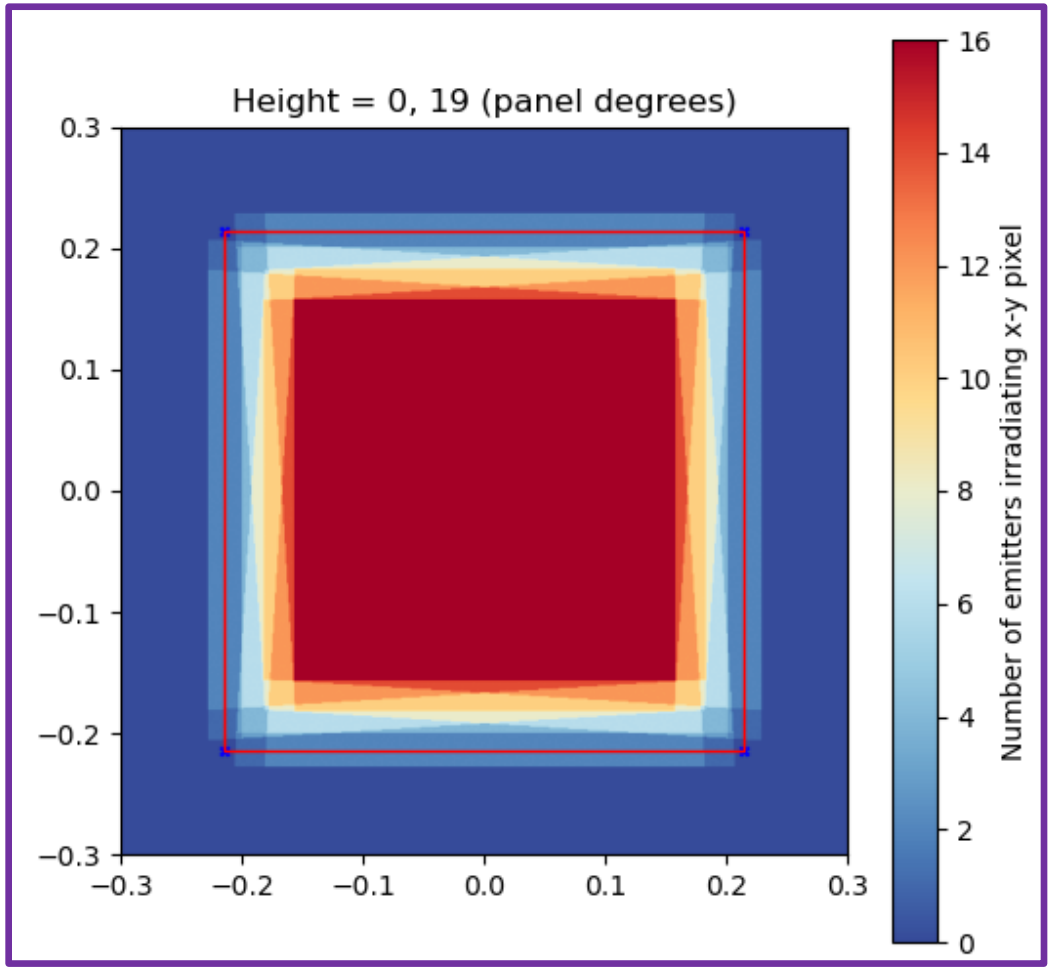

# The EAGLE simulations

EVOLUTION AND ASSEMBLY OF GALAXIES AND THEIR ENVIRONMENTS

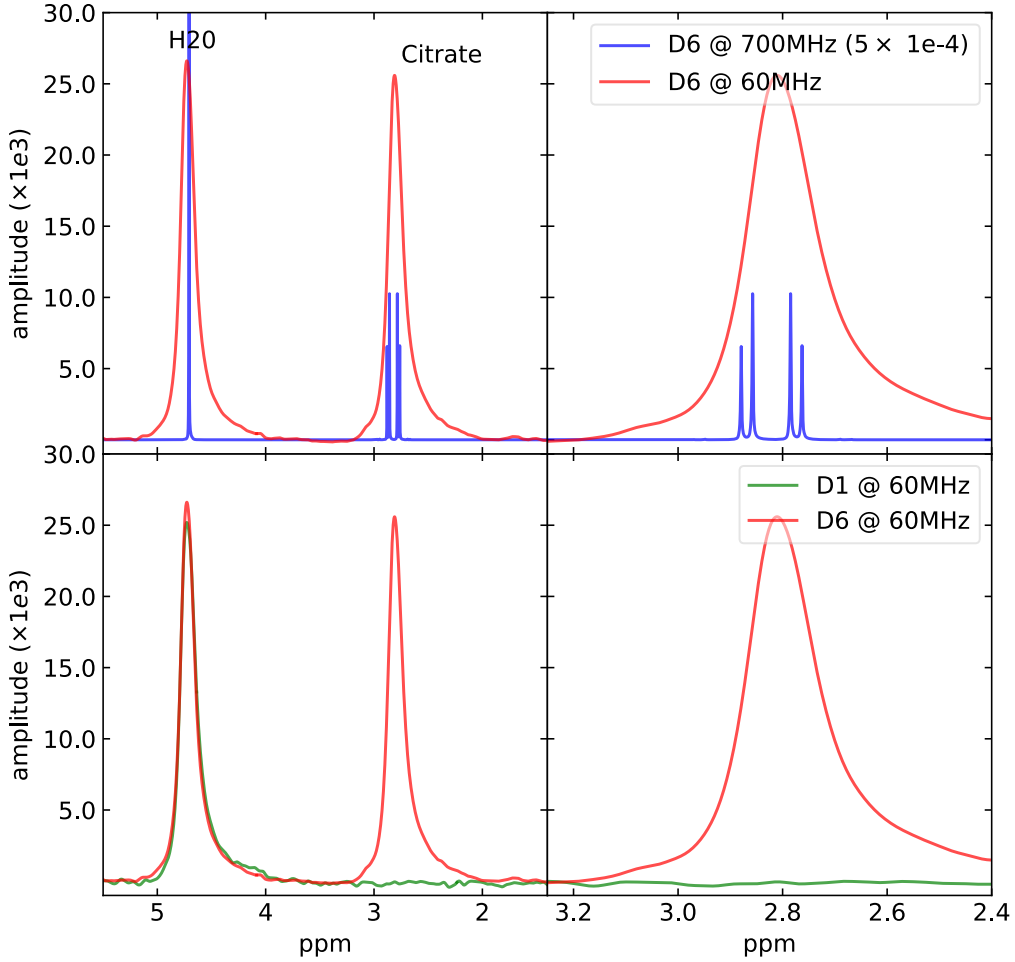
A project of the Virgo consortium

# CURRENT RESEARCH: MEDICAL PHYSICS



# CURRENT RESEARCH: MEDICAL PHYSICS




# MY ROLE HERE

- Conduct my own research, co-supervise several students
- Connect with industry
- Lead some training and organise data science seminar series
- Help you! Ask me for advice with coding, placements, paper writing, living in Liverpool, etc.

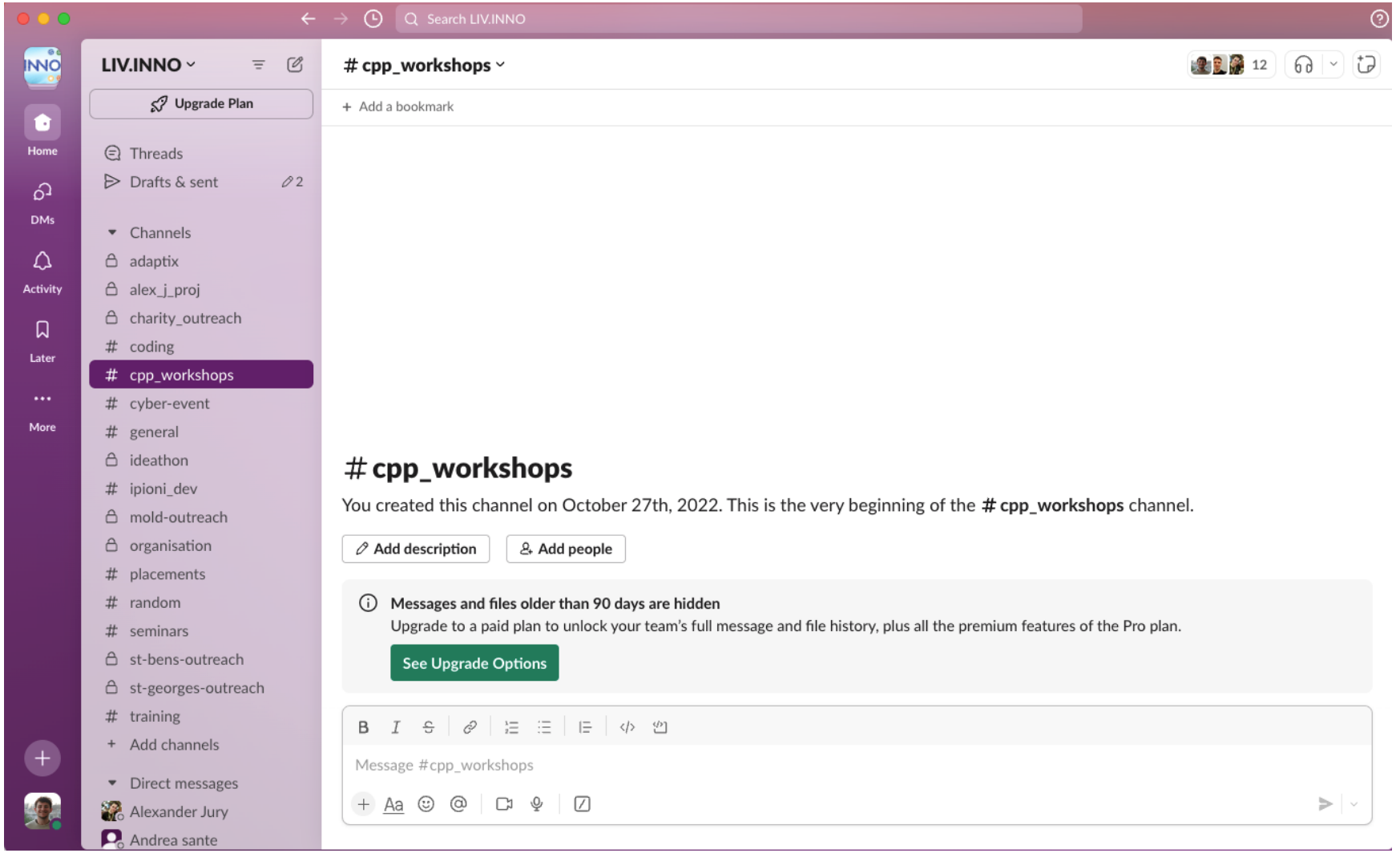
# INTRODUCTIONS

Introduce yourself! (background, coding experience, research interests)

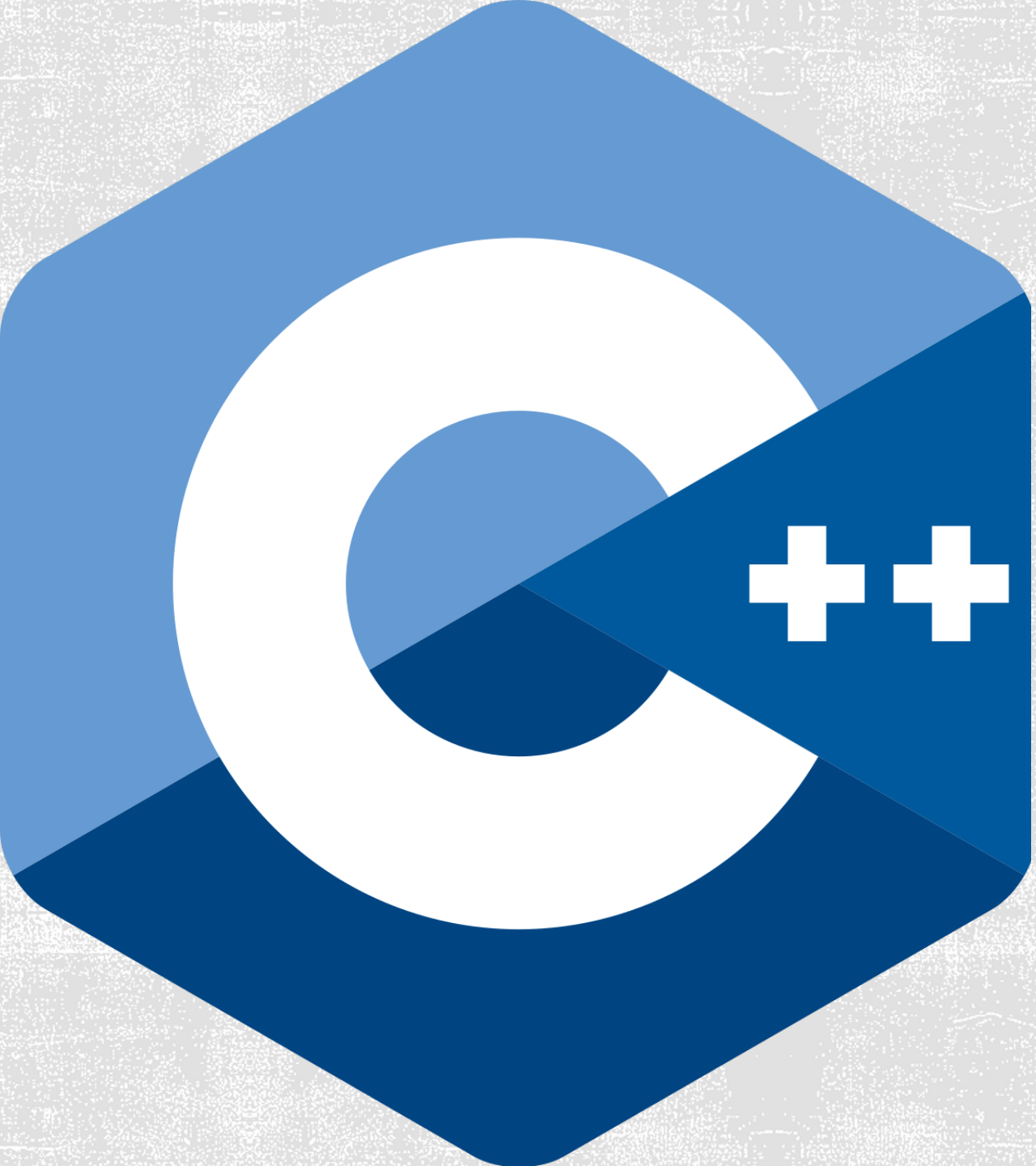
What do you want to get from this PhD, and this course in particular?

What coding challenges might you have over the course of your PhD?

# SLACK CHANNEL – EMAIL A.D.HILL@LIVERPOOL.AC.UK



# WHAT IS C++?



# WHAT IS C++?

- C++ is a superset of the programming language C (so two languages for the price of one!)
- Embodies the philosophy of Object-Oriented Programming
- Extended set of libraries
- Millions of developers worldwide
- Commonly used in conjunction with languages like Python on big projects

# WHAT IS C?

# WHAT IS C?

- Created in the 1970s by Dennis Ritchie and Ken Thompson at Bell Labs
- The first ‘programmer’s language’: general purpose and human-readable
- A middle-level programming language

# PROGRAMMING LANGUAGES

## Low level

- E.g. Assembly language, machine code
- Provides nothing more than access to the machine
- Little (if any) abstraction

## Middle level

- E.g. C, C++
- Provides a user with a concise set of tools, while still offering flexibility with data management

## High level

- E.g. Python, Perl,
- Aims to give the programmer everything they could want
- Highly human-readable and abstract

# FEATURES OF C

- C allows you to manipulate the constituent components of your computer
- No buffer between programme and hardware
- Requires the user to define routines for performing high-level operations

# WHY C++?

- A better way to manage greater complexity
- As the required tasks of computers become more complex, a higher level of abstraction is required
- Object oriented programming is a way to achieve this
- Little sacrifice in the efficiency and flexibility of C

# C++ HISTORY

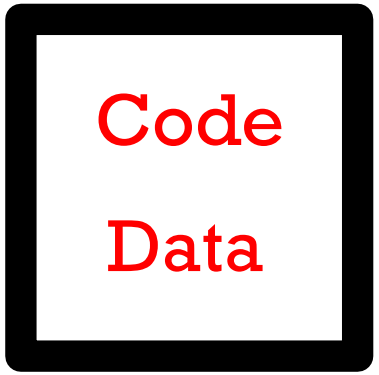
- Created by Bjarne Stroustrup in 1979 at Bell Labs
- Originally called ‘C with Classes’
- Renamed C++ due to the ++ increment operator (a = 0, a++, print(a) => 1)

# OBJECT ORIENTED PROGRAMMING

- Idea: decompose a problem into constituent parts
- Each component has its own rules and data
- Complexity is reduced
- Three characteristics: encapsulation, polymorphism, and inheritance

# ENCAPSULATION: BETTER CONTROL

- Programmes are made of code and data
- Encapsulation relates to the binding of a set of code and data in a black box
- This **object** can be private or public
- The complexity of the code and data is hidden from the user: **abstraction**




# POLYMORPHISM: BETTER FLEXIBILITY

- Polymorphism means that objects of different classes can be treated in the same way
- This means that code can work with objects of various types in a consistent way
- Example, knowing how a steering wheel works enables you to drive a car, tractor, bumper car, etc. The specifics of how the wheel interacts with the vehicle takes place at a lower level
- A coding example is only needing one routine to act on separate lists of integers, floats, and characters
- The compiler selects the method

# INHERITANCE: BETTER RE-USE OF CODE

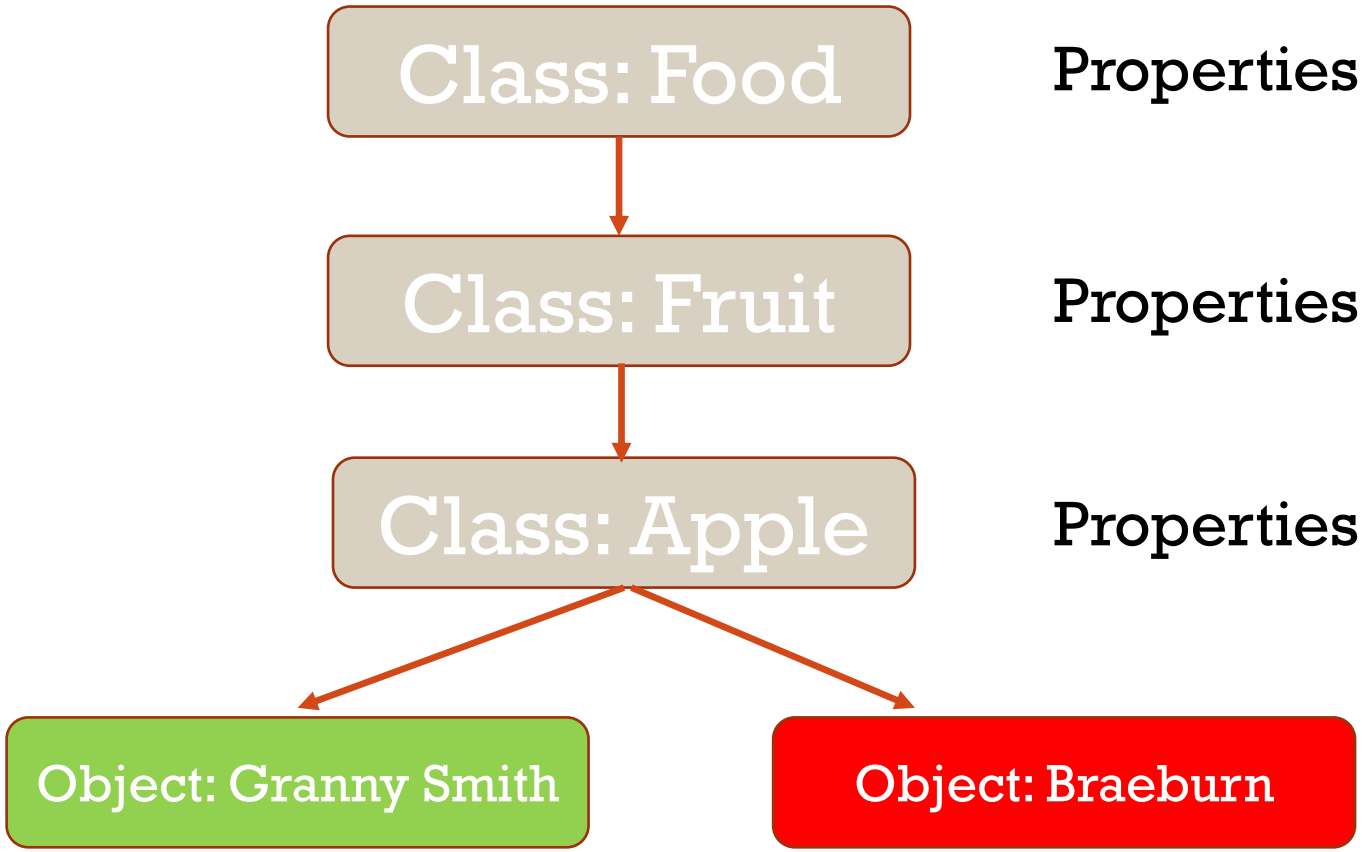
- An object can acquire the properties of another object

- Consider a Braeburn apple object. It has properties {edible, nutritious, seeded, sweet, red}

- Now consider a Granny Smith apple object. It has properties {edible, nutritious, seeded, sweet, green}

- Inheritance allows us to avoid repetition in our work, and provides a way of grouping similar objects together

# INHERITANCE




Properties: Edible, nutritious

Properties: Seeded, plant-based

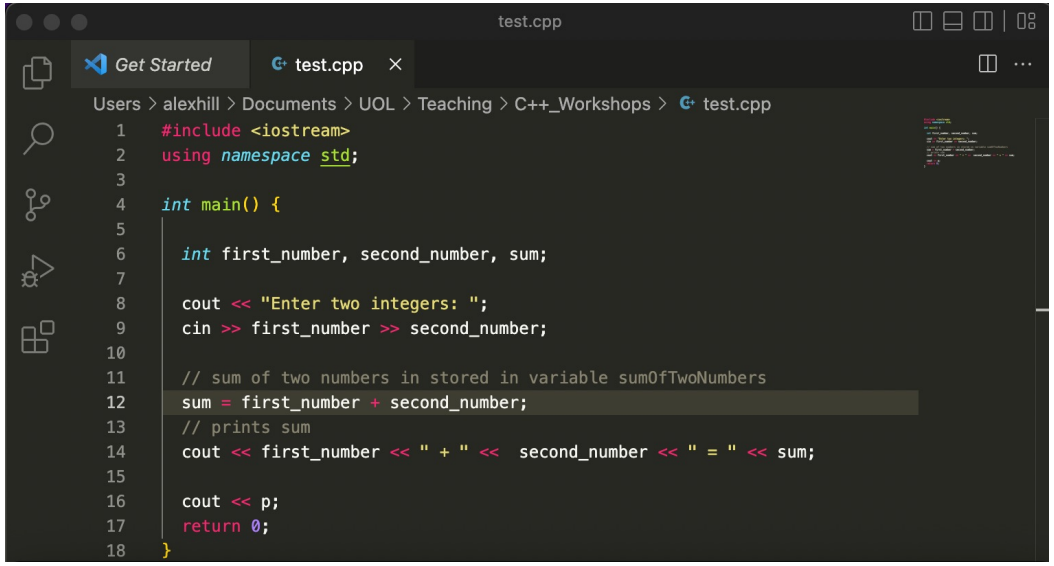
Properties: Sweet, round

# C++ IS FLEXIBLE. YOU CAN USE OOP, BUT YOU DON'T HAVE TO

# GETTING STARTED WITH C++

test.cpp

```
#include <iostream>
using namespace std;

int main() {

  int first_number, second_number, sum;

  cout << "Enter two integers: ";
  cin >> first_number >> second_number;

  // sum of two numbers in stored in variable sumOfTwoNumbers
  sum = first_number + second_number;
  // prints sum
  cout << first_number << " + " <<  second_number << " = " << sum;

  cout << p;
  return 0;
}
```




# IDE – INTEGRATED DEVELOPMENT ENVIRONMENT

- Provides an editor, compiler, and debugger
- I use an IDE (Visual Studio Code) as a fancy text editor, as it enables things like multi-line editing, autocomplete, useful colouring of different syntax

# COMPILERS

Compilers convert the **human-readable source code** that you write into something **readable by the CPU** (e.g. machine code or byte code)

```
#include <iostream>
int main() {
int a = 5;
int b = 10;
int sum = a + b;
std::cout << "The sum of a and b is: " <<
sum << std::endl;
return 0;
 }
```

Compiler

```
.global main
main:
push rbp
mov rbp, rsp
mov DWORD PTR [rbp-20], 5
mov DWORD PTR [rbp-16], 10
mov eax, DWORD PTR [rbp-20]
add eax, DWORD PTR [rbp-16]
mov DWORD PTR [rbp-12], eax
lea rdi, [rip+15] # Address of the string
"The sum of a and b is: "
call std::operator<<(std::basic_ostream<char,
std::char_traits<char> >&, char const*)
mov rax, QWORD PTR [rip+26] # Address of std::cout
mov rsi, QWORD PTR [rip+38] # Address of the variable 'sum'
mov rdi, rax
call std::basic_ostream<char, std::char_traits<char>
>::operator<<(int)
mov esi, 10
mov rdi, rax
call std::basic_ostream<char, std::char_traits<char>
>::operator<<(int)
mov esi, 10
mov rdi, rax
call std::basic_ostream<char, std::char_traits<char>
>::operator<<(int)
mov esi, 10
mov rdi, rax
call std::basic_ostream<char, std::char_traits<char>
>::operator<<(int)
mov rdi, rax
call std::basic_ostream<char, std::char_traits<char>
>::operator<<(char)
mov rdi, rax
call std::basic_ostream<char, std::char_traits<char>
>::operator<<(std::basic_ostream<char, std::char_traits<char>
>&(*)(std::basic_ostream<char, std::char_traits<char> >&))
mov eax, 0
pop rbp
ret
```

# COMPILERS

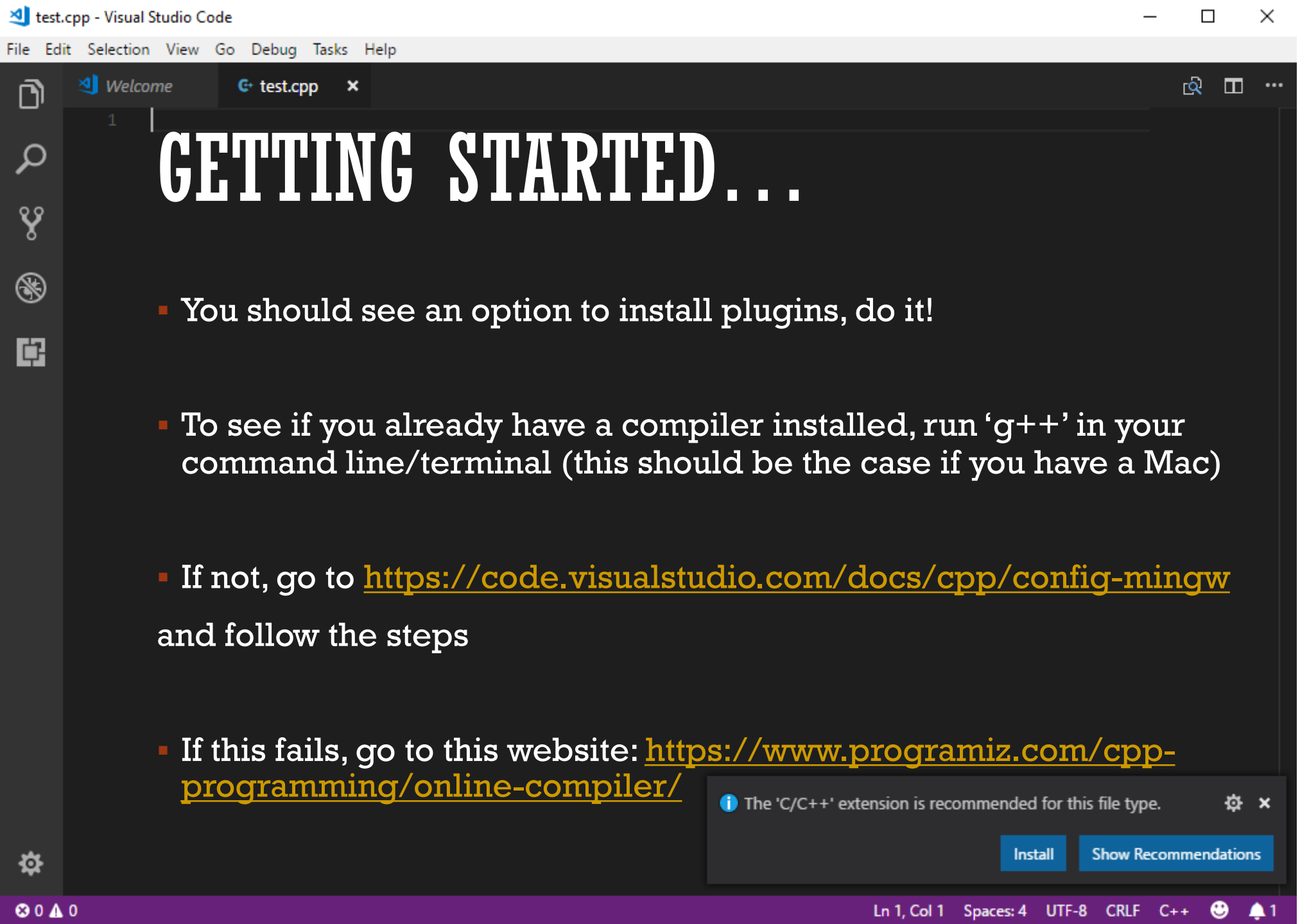
- Compilers convert the **human-readable source code** that you write into something **readable by the CPU** (e.g. machine code or byte code)
- I will use g++ (GNU C++)
- In order to run a C++ script, you first compile it
- `$ g++ -o name-you-want-to-give test.cpp`
- `$ ./name-you-want-to-give`

# EXAMPLE

```
(base) alexhill at Alexs-Air in ~/Documents/UOL/Teaching/C++_Workshops/Workshops
$ ls
test.cpp
(base) alexhill at Alexs-Air in ~/Documents/UOL/Teaching/C++_Workshops/Workshops
$ g++ -o ws_ex1 test.cpp
(base) alexhill at Alexs-Air in ~/Documents/UOL/Teaching/C++_Workshops/Workshops
$ ls
test.cpp        ws_ex1
(base) alexhill at Alexs-Air in ~/Documents/UOL/Teaching/C++_Workshops/Workshops
$ ./ws_ex1
Enter two integers: 3 11
3 + 11 = 14
(base) alexhill at Alexs-Air in ~/Documents/UOL/Teaching/C++_Workshops/Workshops
$
```

# GETTING STARTED…

- Download an IDE, I suggest Visual Studio Code, which supports C++, C#, Java, Python, and others
  - https://code.visualstudio.com/download
- Create a new directory somewhere called Workshops, inside which create a file called ‘test.cpp’
- In Code: File > Open Workspace… > Open ‘Workshops’ > Open test.cpp

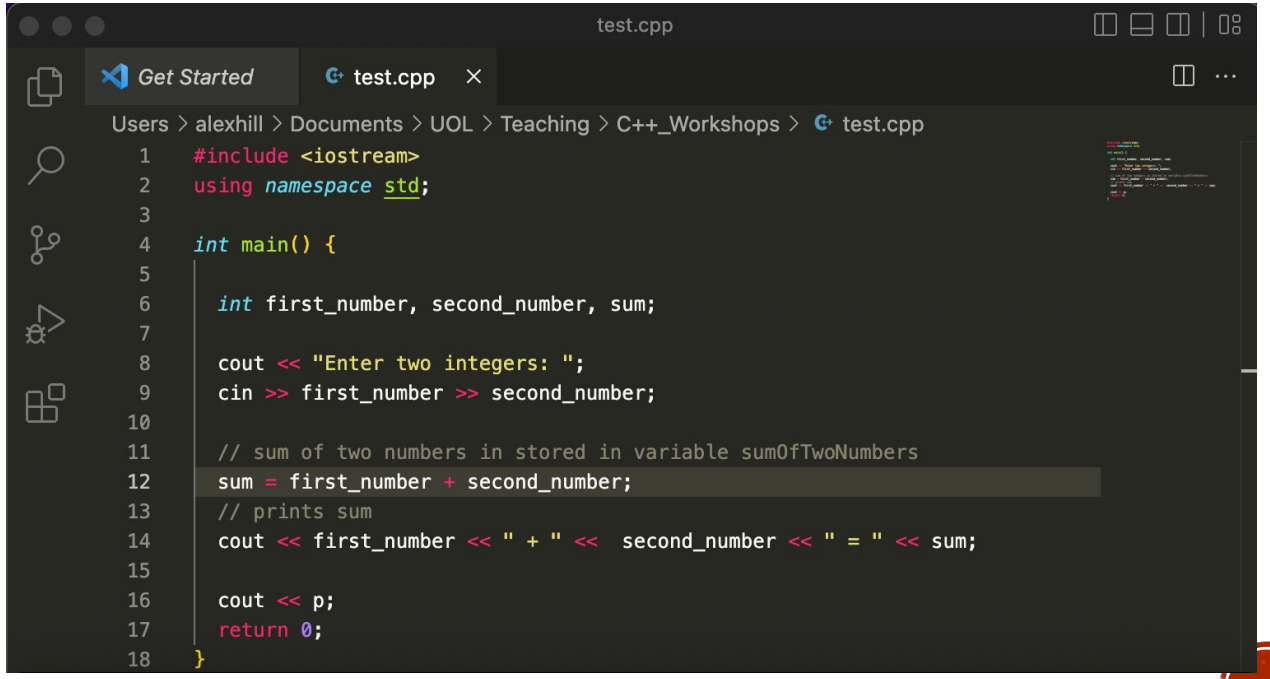

test.cpp - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
Welcome
test.cpp
1
GETTING STARTED...
You should see an option to install plugins, do it!
To see if you already have a compiler installed, run 'g++' in your command line/terminal (this should be the case if you have a Mac)
If not, go to https://code.visualstudio.com/docs/cpp/config-mingw
and follow the steps
If this fails, go to this website: https://www.programiz.com/cpp-programming/online-compiler/
The 'C/C++' extension is recommended for this file type.
Install
Show Recommendations
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF C++

# GETTING STARTED...

- We'll start off with a simple code that creates an executable which takes two numbers and prints out the sum

- Or copy from

https://alex-hill94.github.io/docs/PageOne.html



# RUNNING THE SCRIPT

- Navigate to the Workshops folder in your terminal (or cmd line)
- Or https://www.programiz.com/cpp-programming/online-compiler/
- Try compile the script: `g++ -o name-you-want-to-give test.cpp`
- If using a Mac, run compiled script with: `./name-you-want-to-give`
- If using Windows, run compiled script with: `name-you-want-to-give.exe`

# CONGRATULATIONS/COMMISERATIONS!

# WHAT'S HAPPENING HERE?

```cpp
#include <iostream>
using namespace std;

int main() {

  int first_number, second_number, sum;

  cout << "Enter two integers: ";
  cin >> first_number >> second_number;

  // sum of two numbers in stored in variable sumOfTwoNumbers
  sum = first_number / second_number;
  // prints sum
  cout << first_number << " + " <<  second_number << " = " << sum << "\n";
  return 0;

}
```

# WHAT'S HAPPENING HERE?

```
1    #include <iostream>
```

C++ defines several **headers**, containing information useful or necessary to your programme

For this examples, the header <iostream> is needed to support the C++ I/O system

This header is included with your compiler, no further downloads are needed

Headers are included with the **#include** command

# WHAT'S HAPPENING HERE?

```
2    using namespace std;
```

This tells the compiler to use the **std namespace**

A namespace creates a declarative region within which programme elements can be placed

Elements declared in one namespace are separate from those declared in another

They help the organisation of large programmes

The **std** namespace is the entire Standard C++ library

# WHAT'S HAPPENING HERE?

```
4    int main() {
```

**main()** is the only  function which must
be included in every C++ programme

This is where the programme execution begins

**{ }** indicates the start and end of the **main()** functions code

**int** specifies the type of data that **main()** will return (integer)

# WHAT'S HAPPENING HERE?

```
6       int first_number, second_number, sum;
```

This defines three variables which will be called within the **main()** function.

They are defined to have an *integer* data type

Notice that all C++ statements end with a semicolon

# WHAT'S HAPPENING HERE?

```
8       cout << "Enter two integers: ";
```

This is a console output statement

It causes 'Enter two integers' to be printed on the screen

This is achieved with the output operator: **<<**

**cout** is a pre-defined identifier which stands for console output

**"Enter two integers:"** is a string, identified with double quotes

# WHAT'S HAPPENING HERE?

```
9       cin >> first_number >> second_number;
```

This is a console input statement

Following the previous statement, the user is prompted to enter two integers into the terminal

This is achieved with the input operator: **<<**

**cout** is a pre-defined identifier which stands for console input

The previously defined variables **first_number** & **second_number** are assigned the values inputted

# WHAT'S HAPPENING HERE?

```
11      // sum of two numbers in stored in variable sumOfTwoNumbers
```

This is a single line comment

// tells the compiler to ignore this sentence

These are used by developers to comment on code

In this case, it tells the reader what the function is doing

# WHAT'S HAPPENING HERE?

```
12      sum = first_number + second_number;
```

This gives the variable **sum** the value of **first_number + second_number**

This is achieved with the addition operator **+**

# WHAT'S HAPPENING HERE?

```
14      cout << first_number << " + " <<  second_number << " = " << sum << "\n";
```

This console output commands prints the stored variables and some strings

This results in a human-readable text output of the operation undertaken by the program

**"\n"** is a command that tells the console to go to the next line

# WHAT'S HAPPENING HERE?

```
15      return 0;
16
17  }
```

This terminates **main()** and causes it to return a value of 0 to the calling process

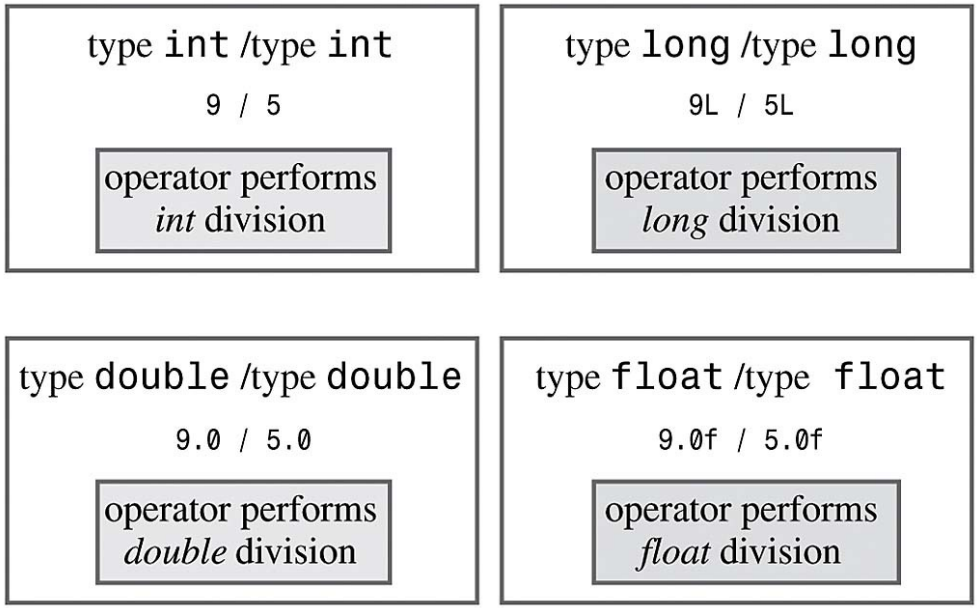
All your programs should return 0 when they terminate normally

**}** is the formal end of the program

# NEXT STEPS

- Play around with this script using other operators (e.g. +, -, /, *) to get a feel for how they work

- Use different data types (e.g. int, float) and see how the operators interact with them



# NEXT WEEK(S)

- Data types
- For loops
- Functions
- Plotting data

# THANKS!