Imperial College London

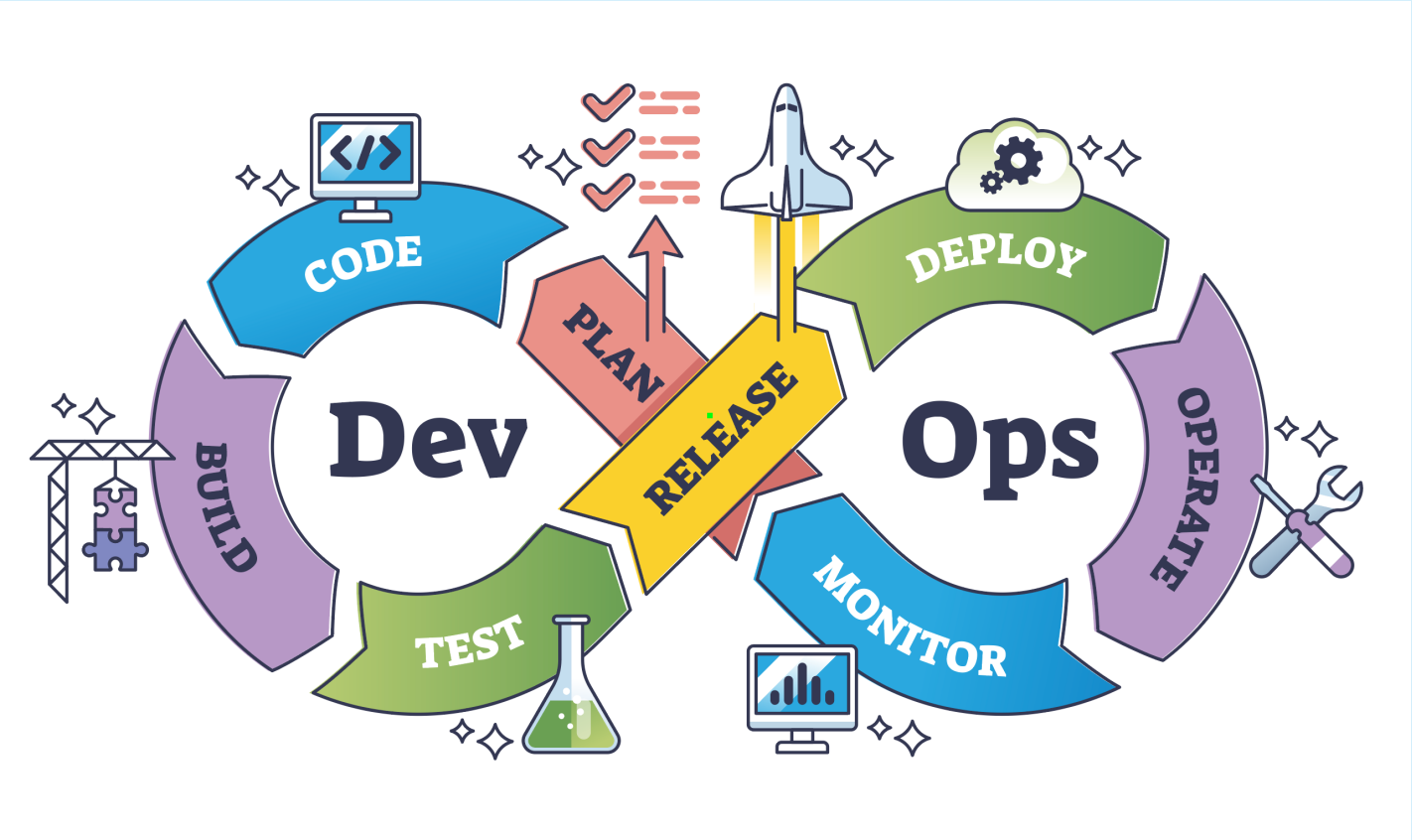
Advanced Programming

Overview

Adriana Paluszny

[A few of the slides are adapted from the original Bjarne Stroustrup slides on C++]

Programming



"Our civilization runs on software" Bjarne Stroustrup

Imperial College London

Advanced Programming Challenges

- Multiple languages
- Precision requirements
- Performance requirements
- Hardware specific
- Big Data
- Larger Scale Projects
- Larger Teams
- Accelerated timelines
- Time sensitive

Imperial College London

Advanced Programming Challenges

- Multiple languages
- Precision requirements
- Performance requirements
- Hardware specific
- Big Data
- Larger Scale Projects
- Larger Teams
- Accelerated timelines
- Time sensitive

C++ runs on hardware such as PCs, mobile devices, web servers, embedded systems (microcontrollers, industrial, automotive systems, gaming consoles, GPUs, IoT)

C++ allows for direct hardware control

C++ allows to control every stage of development

C++ is built for performance

This course

- Learning objectives
  - Fundamental programming concepts
  - Key useful techniques
  - Basic Standard C++ facilities
  - STL Library
  - Basic Memory Management
- After the course, you will ***not*** (yet) be
  - A C++ language expert
  - An expert user of advanced libraries
  - However, you will be on your way!

C++: when performance is required

Imperial College London

List of Topics

- Objects and Types
- Definitions and Declarations
- Scope, Functions, Namespaces
- Classes, Structs, Enumerations
- Inheritance and Polymorphism
- Errors, Exceptions and Debugging
- Pointers and Memory Management
- Templates and STL Library
- Functors
- Streams and Files

The Means

- Lectures
  - Do Attend
  - In Person (alternate between classrooms)
  - Slides/Exercises/Breaks
  - Try to follow the code presented or written in class, ask and interrupt if you want more/other details or don't follow or understand!
  - You can ask questions in the chat.
- Notes/Chapters
  - Bjarne Stroustrup: Programming -- Principles and Practice Using C++ (most examples taken from here)
  - Feedback is welcome (typos, suggestions, etc.)
- Assignments + Homeworks
  - "That's where the most fun and the best learning takes place"
  - One Group Project 50%
  - Individual Coursework: two deliverables (week1+2): 50%
  - Quiz (Self-Assessment/no mark)

Cooperate on Learning

Except for the work you hand in as individual contributions, we ***strongly*** encourage you to collaborate and help each other

Pair programming: structured/unstructured is often of benefit

Don’t copy code from the internet, write it yourself

Feedback request

- Please mail questions and constructive comments to
  apaluszn@imperial.ac.uk or thomas.davison@imperial.ac.uk
  Or any of the teaching or GTA team

- Lead GTA: Ellya Kanimova

- Mon-Tue, Thu-Fri afternoons, GTA sessions
- Group Project Week (Week 3)

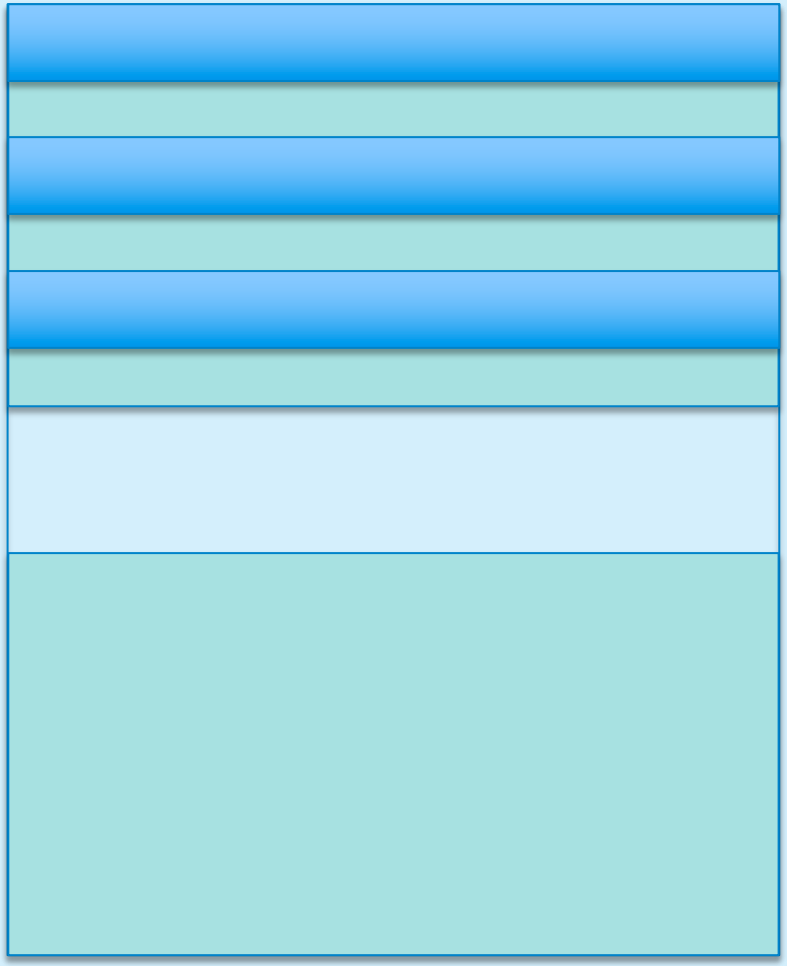
- Your feedback is always be appreciated

Schedule: Lectures

- Introduction
- Declarations & Definitions
- Compiling & Debugging
- Pointers & Memory Management
- Object Types, Values & Classes
- STL Overview
- Functions, Functors & Profiling
- Streams & Files
- STL Operators, Iterators, Algorithms
- Inheritance & Polymorphism

A day

- We will aim for 30 min class, 10-20 min exercise loops, plus breaks



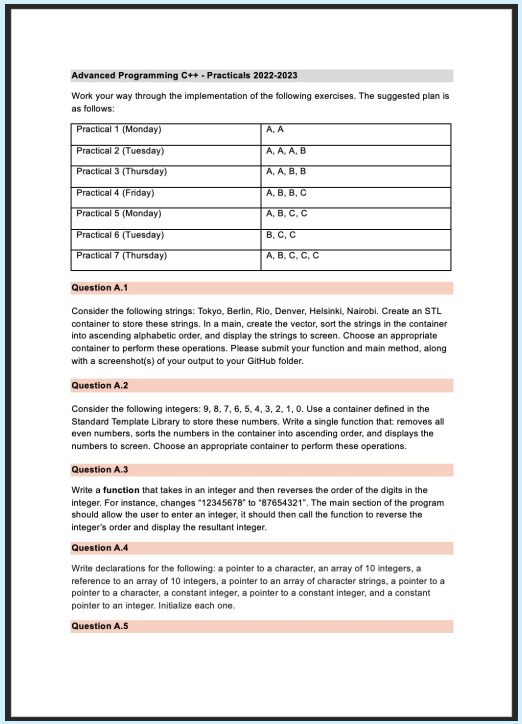
9:30 – 12:00
In person
Lectures

14:00 – 17:00
In person/Teams
GTAs

Imperial College London

Practicals

Practical 1 (Monday)	A, A
Practical 2 (Tuesday)	A, A, A, B
Practical 3 (Thursday)	A, A, B, B
Practical 4 (Friday)	A, B, B, C
Practical 5 (Monday)	A, B, C, C
Practical 6 (Tuesday)	B, C, C
Practical 7 (Thursday)	A, B, C, C, C



Advanced Programming C++ - Practicals 2022-2023

Work your way through the implementation of the following exercises. The suggested plan is as follows:

Practical 1 (Monday)	A, A
Practical 2 (Tuesday)	A, A, A, B
Practical 3 (Thursday)	A, A, B, B
Practical 4 (Friday)	A, B, B, C
Practical 5 (Monday)	A, B, C, C
Practical 6 (Tuesday)	B, C, C
Practical 7 (Thursday)	A, B, C, C, C

Question A.1

Consider the following strings: Tokyo, Berlin, Rio, Denver, Helsinki, Nairobi. Create an STL container to store these strings. In a main, create the vector, sort the strings in the container into ascending alphabetic order, and display the strings to screen. Choose an appropriate container to perform these operations. Please submit your function and main method, along with a screenshot(s) of your output to your GitHub folder.

Question A.2

Consider the following integers: 9, 8, 7, 6, 5, 4, 3, 2, 1, 0. Use a container defined in the Standard Template Library to store these numbers. Write a single function that: removes all even numbers, sorts the numbers in the container into ascending order, and displays the numbers to screen. Choose an appropriate container to perform these operations.

Question A.3

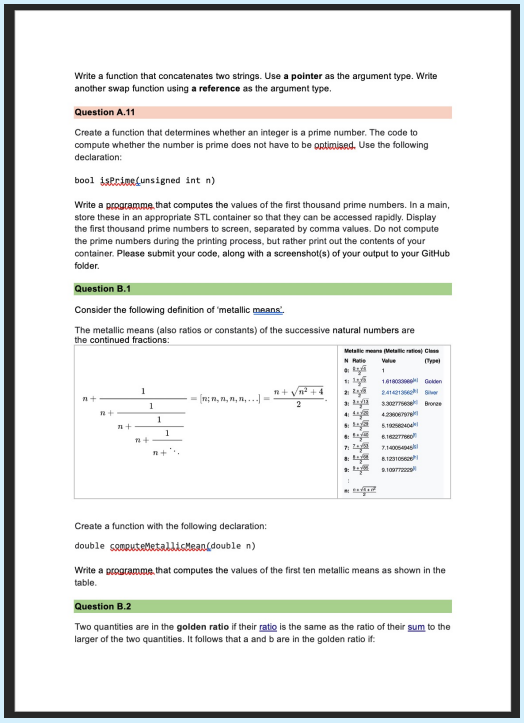
Write a **function** that takes in an integer and then reverses the order of the digits in the integer. For instance, changes "12345678" to "87654321". The main section of the program should allow the user to enter an integer, it should then call the function to reverse the integer's order and display the resultant integer.

Question A.4

Write declarations for the following: a pointer to a character, an array of 10 integers, a reference to an array of 10 integers, a pointer to an array of character strings, a pointer to a pointer to a character, a constant integer, a pointer to a constant integer, and a constant pointer to an integer. Initialize each one.

Question A.5

...



Write a function that concatenates two strings. Use **a pointer** as the argument type. Write another swap function using **a reference** as the argument type.

Question A.11

Create a function that determines whether an integer is a prime number. The code to compute whether the number is prime does not have to be optimised. Use the following declaration:

```
bool isPrime(unsigned int n)
```

Write a programme that computes the values of the first thousand prime numbers. In a main, store these in an appropriate STL container so that they can be accessed rapidly. Display the first thousand prime numbers to screen, separated by comma values. Do not compute the prime numbers during the printing process, but rather print out the contents of your container. Please submit your code, along with a screenshot(s) of your output to your GitHub folder.

Question B.1

Consider the following definition of 'metallic means'.

The metallic means (also ratios or constants) of the successive natural numbers are the continued fractions:

$$n + \cfrac{1}{n + \cfrac{1}{n + \cfrac{1}{n + \ddots}}} = [n; n, n, n, \ldots] = \frac{n + \sqrt{n^2 + 4}}{2}.$$

N	Ratio	Value	(Type)
0:	$\frac{0+\sqrt{0}}{2}$	1	
1:	$\frac{1+\sqrt{5}}{2}$	1.6180339887[91]	Golden
2:	$\frac{2+\sqrt{8}}{2}$	2.4142135623[73]	Silver
3:	$\frac{3+\sqrt{13}}{2}$	3.3027756377[32]	Bronze
4:	$\frac{4+\sqrt{20}}{2}$	4.2360679775[48]	
5:	$\frac{5+\sqrt{29}}{2}$	5.1925824035[94]	
6:	$\frac{6+\sqrt{40}}{2}$	6.1622776601[68]	
7:	$\frac{7+\sqrt{53}}{2}$	7.1400549446[40]	
8:	$\frac{8+\sqrt{68}}{2}$	8.1231056256[17]	
9:	$\frac{9+\sqrt{85}}{2}$	9.1097722286[46]	
⋮			
n:	$\frac{n+\sqrt{n^2+4}}{2}$		

Create a function with the following declaration:

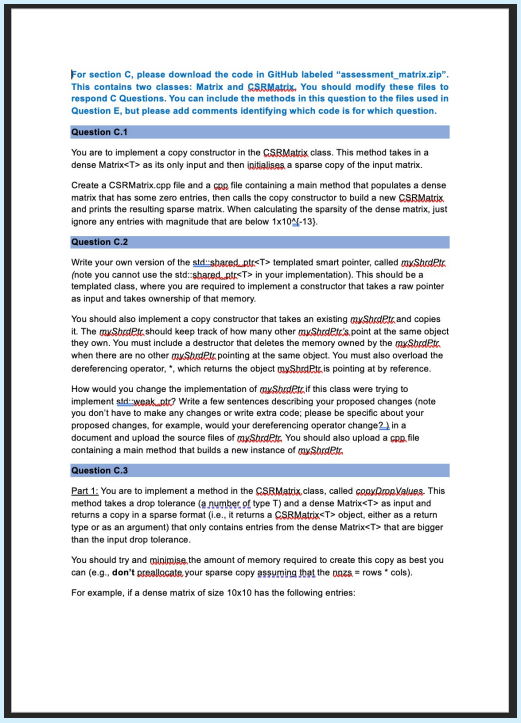
```
double computeMetallicMean(double n)
```

Write a programme that computes the values of the first ten metallic means as shown in the table.

Question B.2

Two quantities are in the **golden ratio** if their ratio is the same as the ratio of their sum to the larger of the two quantities. It follows that a and b are in the golden ratio if:

...



For section C, please download the code in GitHub labeled "assessment_matrix.zip". This contains two classes: Matrix and CSRMatrix. You should modify these files to respond C Questions. You can include the methods in this question to the files used in Question E, but please add comments identifying which code is for which question.

Question C.1

You are to implement a copy constructor in the CSRMatrix class. This method takes in a dense Matrix<T> as its only input and then initialises a sparse copy of the input matrix.

Create a CSRMatrix.cpp file and a cpp file containing a main method that populates a dense matrix that has some zero entries, then calls the copy constructor to build a new CSRMatrix and prints the resulting sparse matrix. When calculating the sparsity of the dense matrix, just ignore any entries with magnitude that are below 1x10^(-13).

Question C.2

Write your own version of the std::shared_ptr<T> templated smart pointer, called *myShrdPtr*. (note you cannot use the std::shared_ptr<T> in your implementation). This should be a templated class, where you are required to implement a constructor that takes a raw pointer as input and takes ownership of that memory.

You should also implement a copy constructor that takes an existing *myShrdPtr* and copies it. The *myShrdPtr* should keep track of how many other *myShrdPtr's* point at the same object they own. You must include a destructor that deletes the memory owned by the *myShrdPtr* when there are no other *myShrdPtr* pointing at the same object. You must also overload the dereferencing operator, *, which returns the object myShrdPtr is pointing at by reference.

How would you change the implementation of *myShrdPtr* if this class were trying to implement std::weak_ptr? Write a few sentences describing your proposed changes (note you don't have to make any changes or write extra code; please be specific about your proposed changes, for example, would your dereferencing operator change?) in a document and upload the source files of *myShrdPtr*. You should also upload a cpp file containing a main method that builds a new instance of *myShrdPtr*.

Question C.3

Part 1: You are to implement a method in the CSRMatrix class, called *copyDropValues*. This method takes a drop tolerance (a number of type T) and a dense Matrix<T> as input and returns a copy in a sparse format (i.e., it returns a CSRMatrix<T> object, either as a return type or as an argument) that only contains entries from the dense Matrix<T> that are bigger than the input drop tolerance.

You should try and minimise the amount of memory required to create this copy as best you can (e.g., **don't** preallocate your sparse copy assuming that the nnzs = rows * cols).

For example, if a dense matrix of size 10x10 has the following entries:

Advanced Programming

- What does it mean to be an “advanced programmer”?
- “Competitive programming”

Goals

- Modularity
- Abstraction
- Speed
- Security
- Sustainability
- Efficiency (Hw)
- Longevity
- Growth/Scale

Advanced Programming

Advanced programming involves a wide range of concepts and skills, but here are some of the most important ones:

- **Object-oriented programming (OOP)**: OOP is a programming paradigm that uses objects to represent data and the operations that can be performed on that data. It's an important concept to understand for building large-scale applications, as it helps to organize and manage code more effectively.
- **Data structures and algorithms**: Understanding data structures (such as arrays, lists, and trees) and algorithms (such as sorting and searching) is essential for building efficient and effective programs.
- **Memory management**: Memory management is the process of allocating and deallocating memory in a program. Understanding memory management concepts is essential for building efficient programs that don't run out of memory.

Advanced Programming

- **Testing, debugging and profiling:** Knowing how to write and run tests, how to debug problems in your code, and how to measure its efficiency, is crucial for building high-quality software.
- **Design patterns**: Design patterns are reusable solutions to common software design problems. Learning design patterns can help you write code that is more modular, extensible, and maintainable. {We will mention some of these, but not covered}
- {There is also **Concurrency** or **Parallel Programming**, which will not be covered in this course}

Imperial College London

Why C++ ?

- The purpose of a programming language is to allow you to express your ideas in code
- C++ is the language that most directly allows you to express ideas from the largest number of application areas
- Abstraction and Hardware control
- C++ is the most widely used language in engineering areas
  - Amazon, Google, Spotify, YouTube, Facebook, Twitter, Bing
  - Finance (High Frequency Trading**), Simulation, Medical, Engineering, Energy, Visualisation, Manufacturing, Comms, Games, Mars Rover
  - Adobe Photoshop & Illustrator, Google file system, Bloomberg, Microsoft OS, Microsoft Office, Microsoft Visual Studio, Mozilla Firefox, mySQL

C++

Logo endorsed by the C++ standards committee

Paradigms	Multi-paradigm: procedural, imperative, functional, object-oriented, generic, modular
Family	C
Designed by	Bjarne Stroustrup
Developer	ISO/IEC JTC 1 (Joint Technical Committee 1) / SC 22 (Subcommittee 22) / WG 21 (Working Group 21)
First appeared	1985; 39 years ago
Stable release	C++20 (ISO/IEC 14882:2020) / 15 December 2020; 3 years ago
Preview release	C++23 / 19 March 2023; 11 months ago
Typing discipline	Static, strong, nominative, partially inferred
OS	Cross-platform
Filename extensions	.C, .cc, .cpp, .cxx, .c++, .h, .H, .hh, .hpp, .hxx, .h++ .cppm, .ixx,[1]
Website	isocpp.org

Major implementations

GCC, LLVM Clang, Microsoft Visual C++, Embarcadero C++Builder, Intel C++ Compiler, IBM XL C++, EDG

Influenced by

Ada, ALGOL 68,[2] BCPL,[3] C, CLU,[2] F#,[4] [note 1] ML, Mesa,[2] Modula-2,[2] Simula, Smalltalk[2]

Influenced

Ada 95, C#,[5] C99, Carbon, Chapel,[6] Clojure,[7] D, Java,[8] JS++,[9] Lua,[10] Nim,[11] Objective-C++, Perl, PHP, Python,[12] Rust,[13] Seed7

C++ Programming at Wikibooks

Source Wikipedia

Source Stroustrup/Programming

Who created it?

Bjarne Stroustrup

From Wikipedia, the free encyclopedia

"Stroustrup" redirects here. It is not to be confused with Jakob Stoustrup.

Bjarne Stroustrup (/ˈbjɑːrnə ˈstraʊstrʊp/; Danish: [ˈpjɑːnə ˈstʁʌwˀstʁɔp];[2][3] born 30 December 1950) is a Danish computer scientist, most notable for the creation and development of the C++ programming language.[4] He is a visiting professor at Columbia University, and works at Morgan Stanley as a Managing Director in New York.[5][6][7][8][9]

Contents [hide]

Early life and education [edit]

Stroustrup was born in Aarhus, Denmark. His family was working class, and he went to the local schools.[10]

He attended Aarhus University 1969–1975 and graduated with a master's degree in mathematics and computer science. His interests focused on microprogramming and machine architecture. He learned the fundamentals of object-oriented programming from its inventor, Kristen Nygaard, who frequently visited Aarhus.

In 1979, he received a PhD in computer science from the University of Cambridge,[11] where he was supervised by David Wheeler.[1][12] His thesis concerned communication in distributed computer systems.[13]

Bjarne Stroustrup

Stroustrup in 2010

Born	30 December 1950 (age 70) Aarhus, Denmark
Nationality	Danish
Education	Aarhus University (MSc) University of Cambridge (PhD)
Known for	C++
Awards	Grace Murray Hopper Award (1993) ACM Fellow (1994) IEEE Fellow (1994)

Source Wikipedia

Why C++? (As a programmer)

- C++ is precisely and comprehensively defined by an ISO standard
  - And that standard is almost universally accepted
  - The most recent standard is ISO C++ 2020 (2023 underway)
- C++ is available on almost all kinds of computers/environments
- Programming concepts that you learn using C++ can be used fairly directly in other languages
  - Including C, Java, C#, and (less directly) Fortran

C++ standards[34]

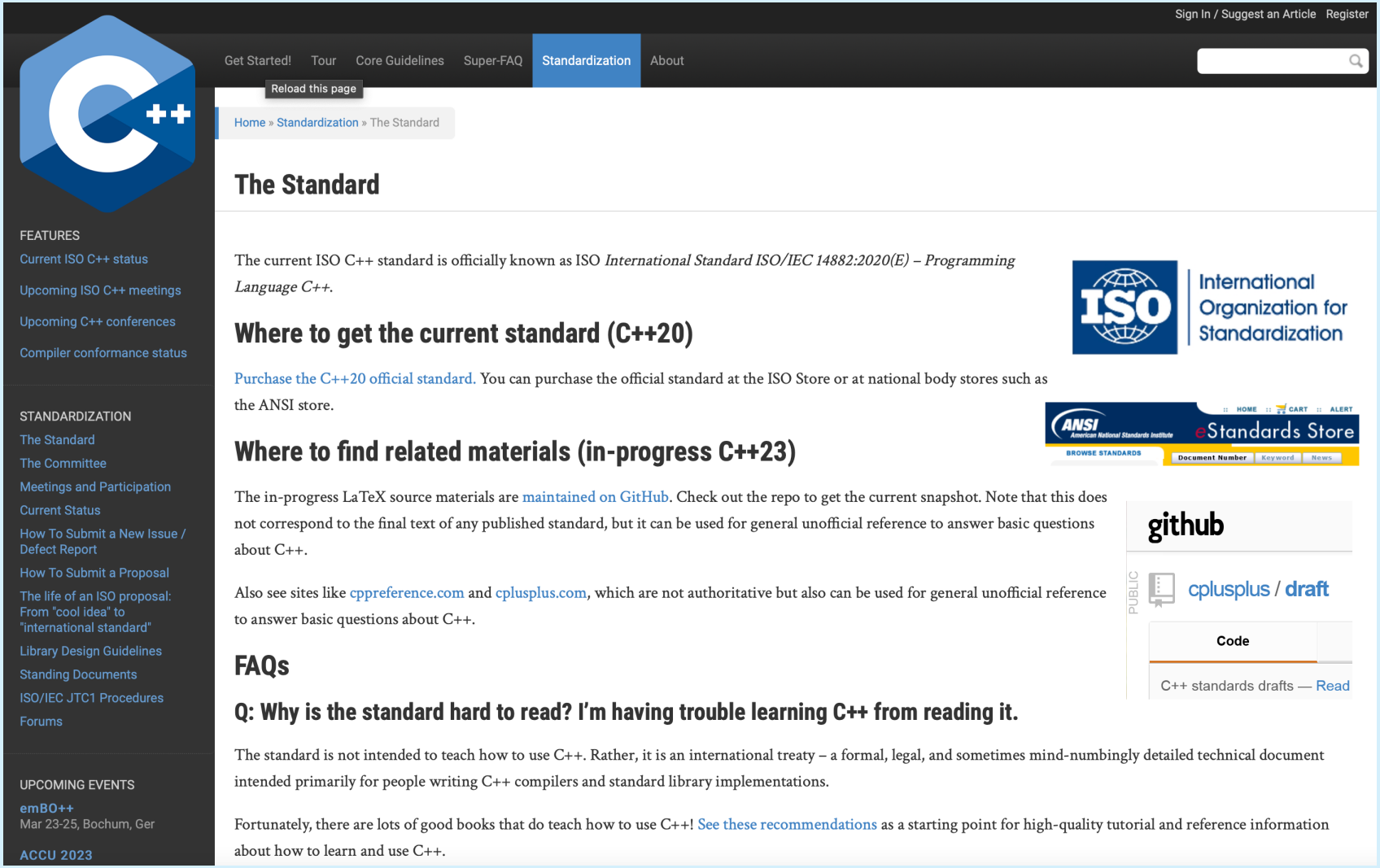
Year	ISO/IEC Standard	Informal name
1998	14882:1998[35]	C++98
2003	14882:2003[36]	C++03
2011	14882:2011[37]	C++11, C++0x
2014	14882:2014[38]	C++14, C++1y
2017	14882:2017[39]	C++17, C++1z
2020	14882:2020[17]	C++20, C++2a
2023		C++23

C++ is changing and evolving all the time, and it is very fast!

Imperial College London

Committee

- ISO C++ WG21
- Standards Committee
- Design C++
- Other companies implement the compilers: Intel, Microsoft, gnu (OA)
- The standard moves faster than the implementation
- https://isocpp.org/std/the-committee



Sign In / Suggest an Article Register

Get Started! Tour Core Guidelines Super-FAQ Standardization About

Reload this page

Home » Standardization » The Standard

FEATURES
Current ISO C++ status
Upcoming ISO C++ meetings
Upcoming C++ conferences
Compiler conformance status

STANDARDIZATION
The Standard
The Committee
Meetings and Participation
Current Status
How To Submit a New Issue / Defect Report
How To Submit a Proposal
The life of an ISO proposal: From "cool idea" to "international standard"
Library Design Guidelines
Standing Documents
ISO/IEC JTC1 Procedures
Forums

UPCOMING EVENTS
emBO++
Mar 23-25, Bochum, Ger
ACCU 2023

The Standard

The current ISO C++ standard is officially known as ISO *International Standard ISO/IEC 14882:2020(E) – Programming Language C++*.

ISO International Organization for Standardization

Where to get the current standard (C++20)

Purchase the C++20 official standard. You can purchase the official standard at the ISO Store or at national body stores such as the ANSI store.

ANSI American National Standards Institute eStandards Store

Where to find related materials (in-progress C++23)

The in-progress LaTeX source materials are maintained on GitHub. Check out the repo to get the current snapshot. Note that this does not correspond to the final text of any published standard, but it can be used for general unofficial reference to answer basic questions about C++.

github
PUBLIC cplusplus / draft
Code
C++ standards drafts — Read

Also see sites like cppreference.com and cplusplus.com, which are not authoritative but also can be used for general unofficial reference to answer basic questions about C++.

FAQs

Q: Why is the standard hard to read? I'm having trouble learning C++ from reading it.

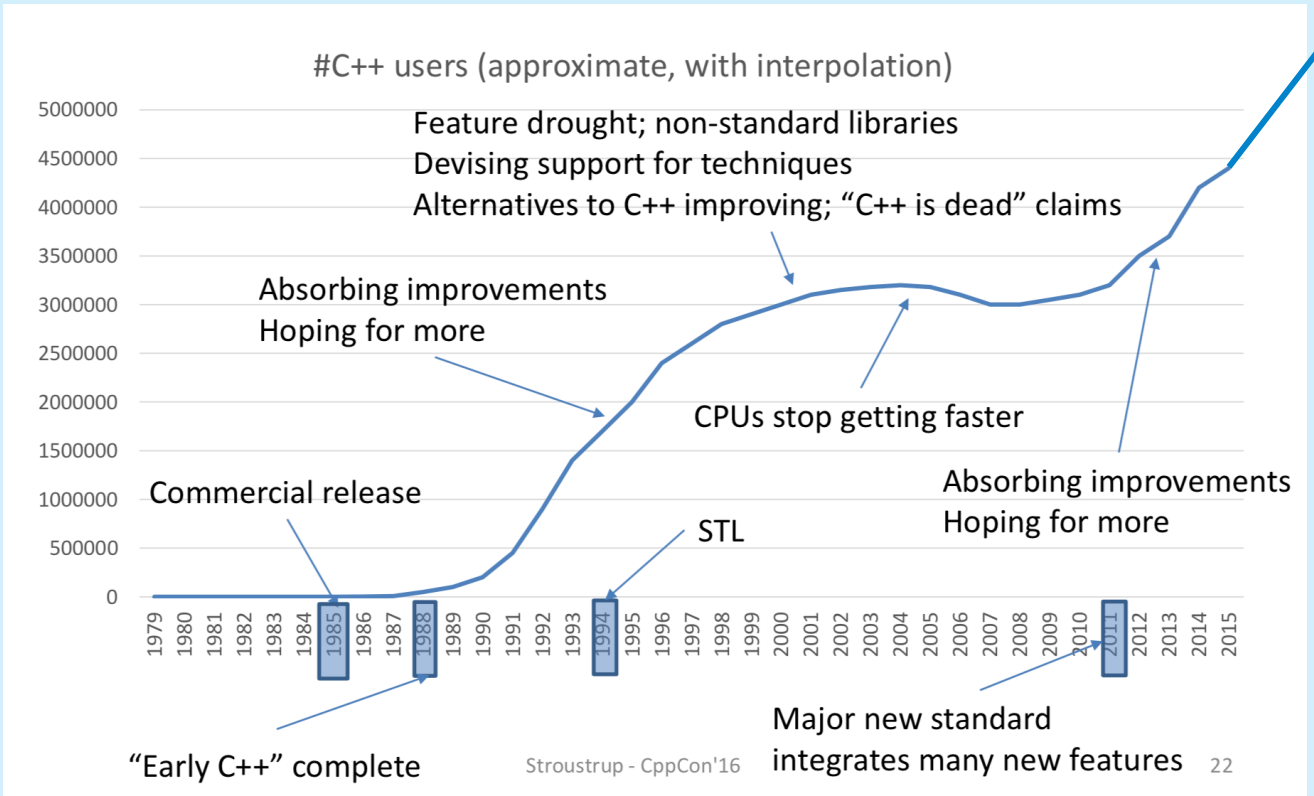
The standard is not intended to teach how to use C++. Rather, it is an international treaty – a formal, legal, and sometimes mind-numbingly detailed technical document intended primarily for people writing C++ compilers and standard library implementations.

Fortunately, there are lots of good books that do teach how to use C++! See these recommendations as a starting point for high-quality tutorial and reference information about how to learn and use C++.

Design Principles

- Evolutionary (Stability, Adaptation)
- Simple things simple
  - Don’t make complicated tasks impossible/unreasonably hard to complete
- Zero-overhead principle
  - Pay for what you use
- Aim high
  - Change the way we think

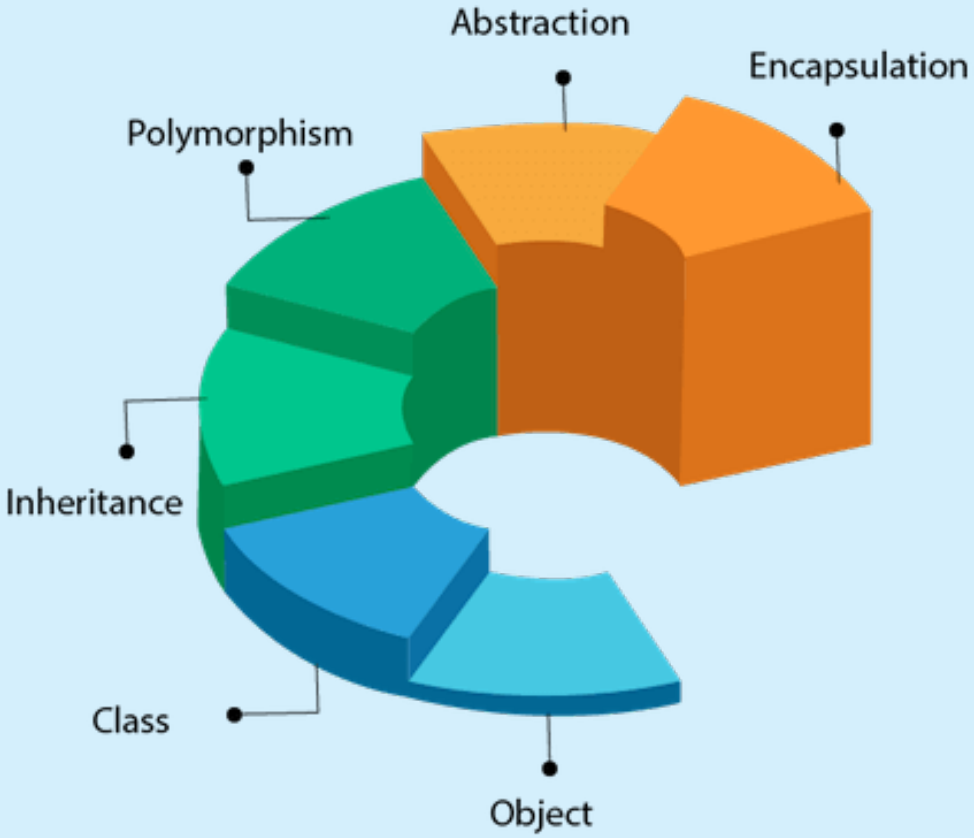
"C++ Success" by Bjarne Stroustrup



Main characteristics of C++

Three main concepts define C++

- Object oriented
- Inheritance
- Polymorphism



Imperial College
London

Python vs C++

Python and C++

- C++ has a stricter syntax
- Learning also requires understanding of inner workings (Mid- to low-level language)
- More prone to human error
- Compiled: executable tailored to platform
- Allows for tailored code
- Allows for substantial optimisation
- Allows control of computer resources
- Fast
- Access to specialised libraries
- Connection to other libraries requires care
- Large systems, Live Systems, Cloud, Gaming, Medical, Banking, Machine Learning …

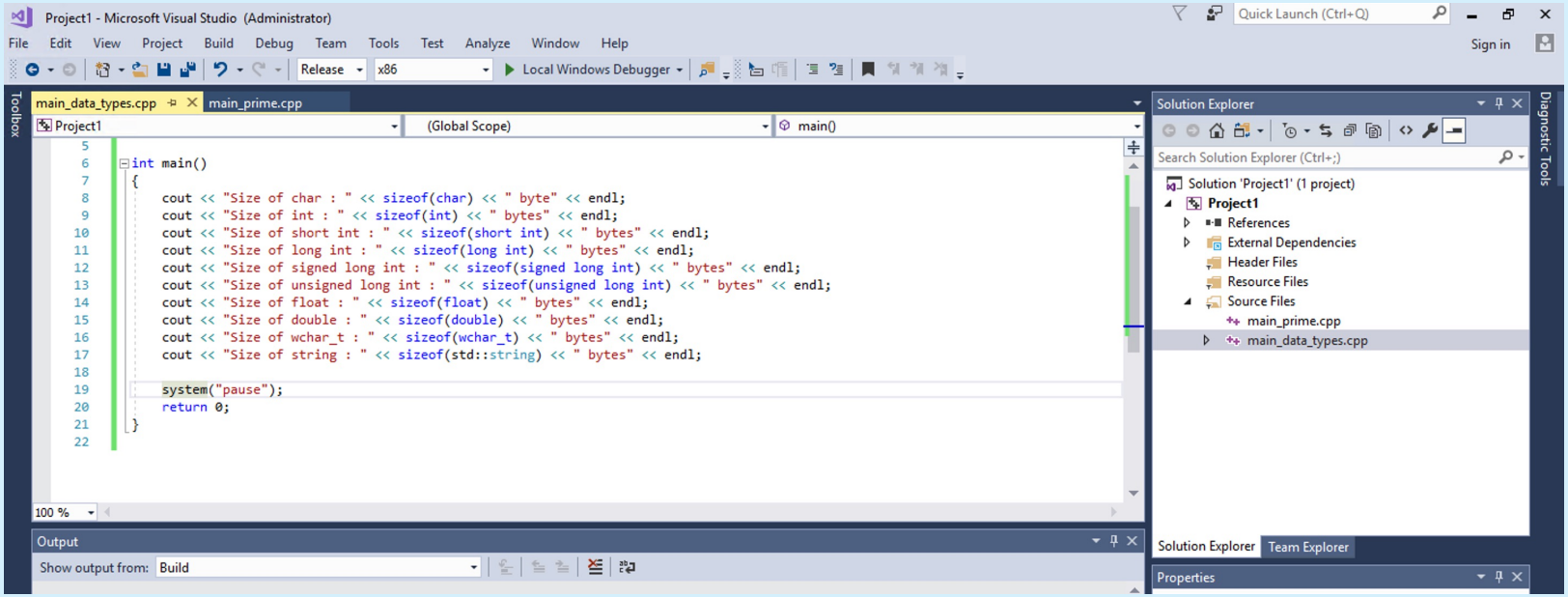
- Python has a syntax designed to make it easy to use
- Quick to learn (high level)
- Less prone to human error
- Interpreted: precompiled and virtual machine
- Code will be run in pre-established ways
- Complex ways to optimise
- Less control
- Slow
- Access to specialized libraries
- Easier to connect to other libraries
- Machine Learning, Testing new concepts, Data Analysis, Research, …

Both can be integrated, both powerful tools in the development stack

An IDE = MSVC Community 2019/2022

Integrated Development Environment (IDE)

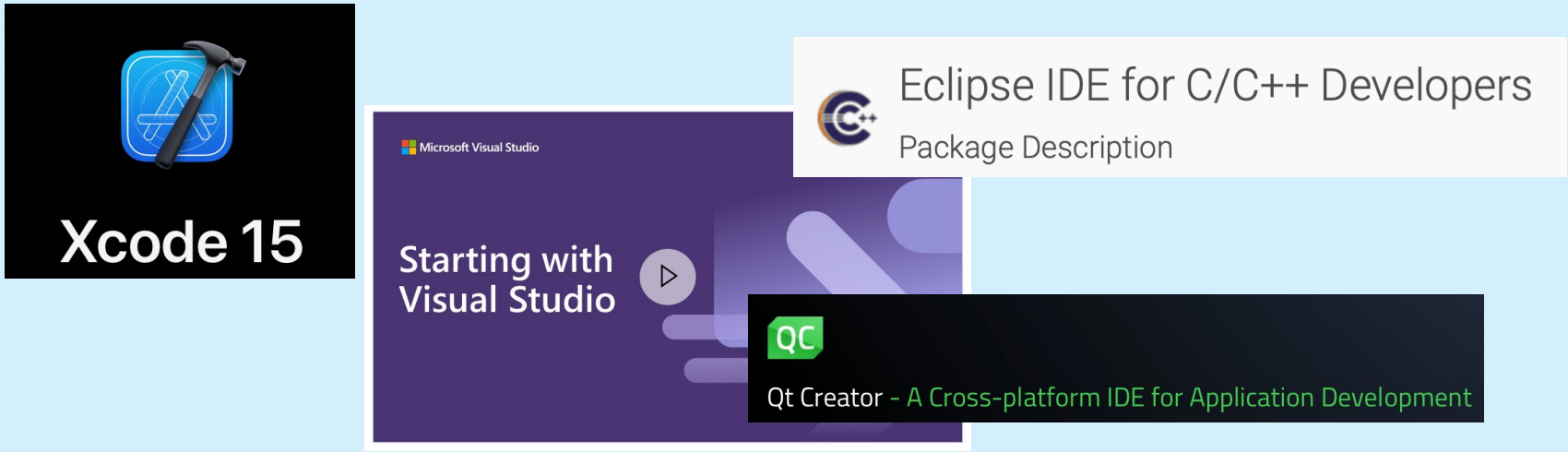
Beyond a file editor it provides support for the construction of the makefiles = Editor + Compiler + Linker + Debugger + Profiler



Many other IDEs and Editors

- Visual Studio Code, Eclipse, Code::Blocks, CodeLite, NetBeans, Qt Creator, Dev C++, Clion, Sublime, Xcode, and others

- And there are also partially integrated environments – basically enriched text editors – with compiling done via command line



Imperial College London

Imperial College London

MSVC Windows

VISUAL STUDIO 2022

MSVC 2022 Windows: Solution vs. Project

- A solution includes many related projects
- Projects can be in different languages

[MSVC++ compiler]

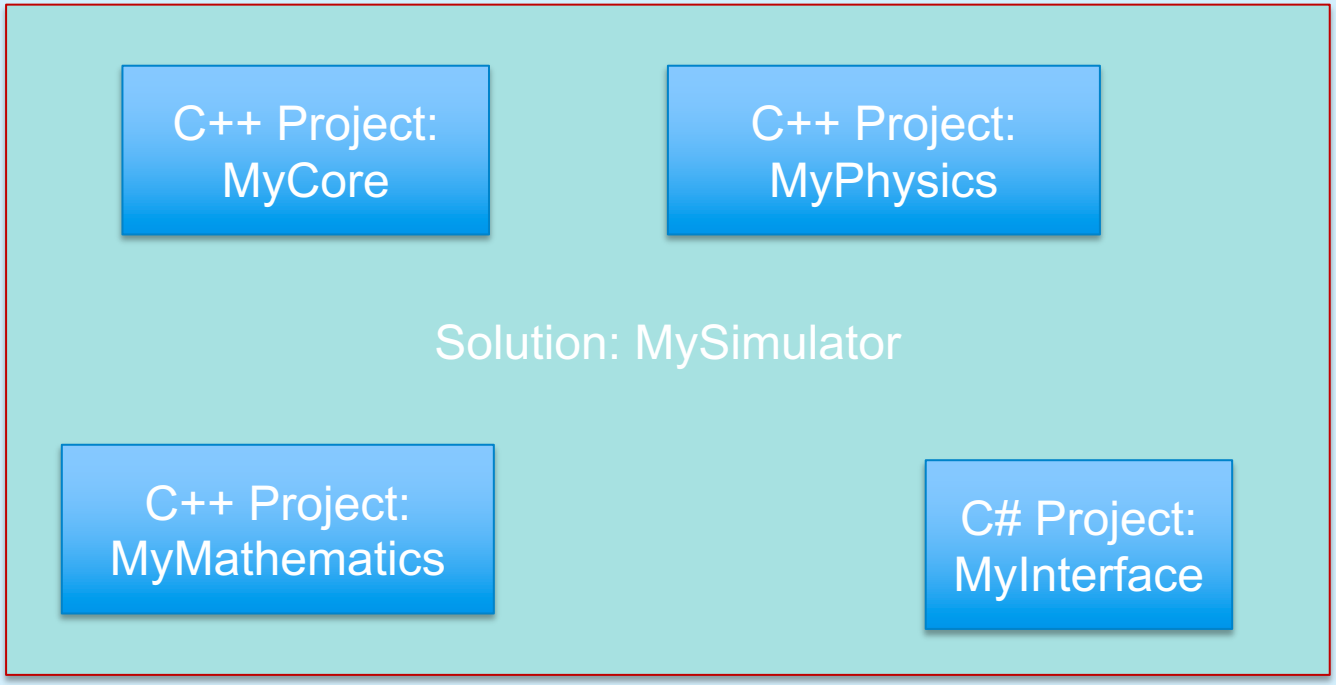
[editor]

[profiler]

[debugger]

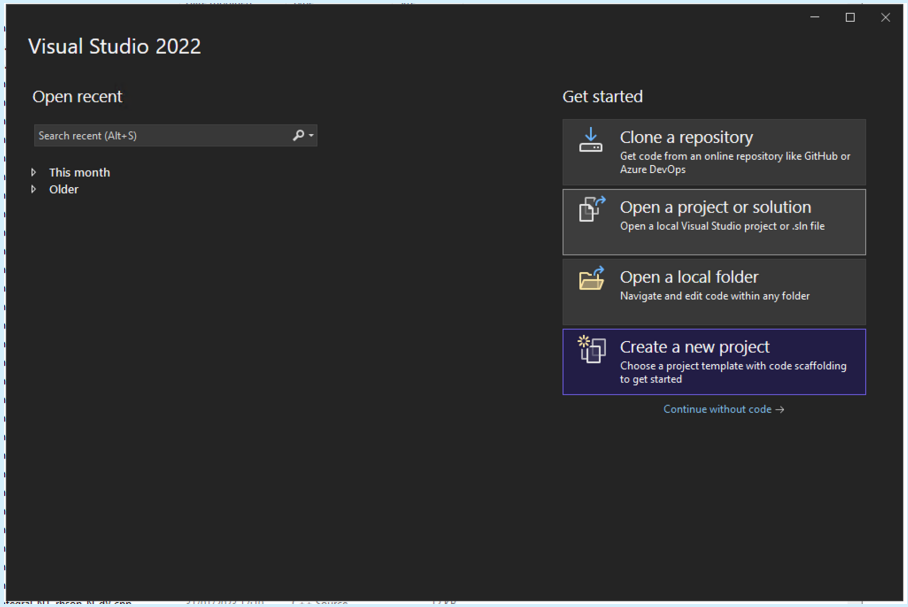
[pro tool]

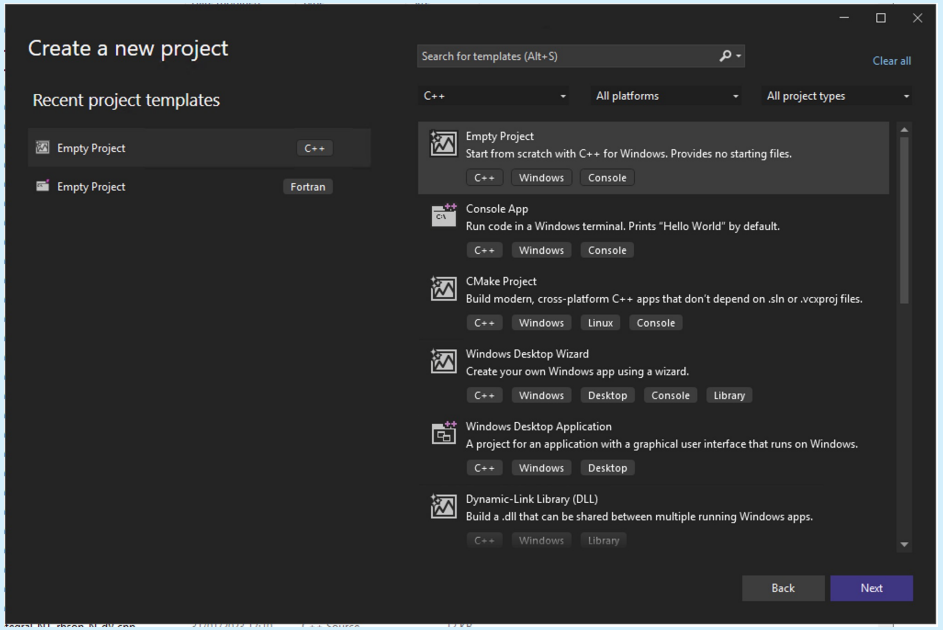
[free]



MSVC Windows

Visual Studio Community 2022: New Project



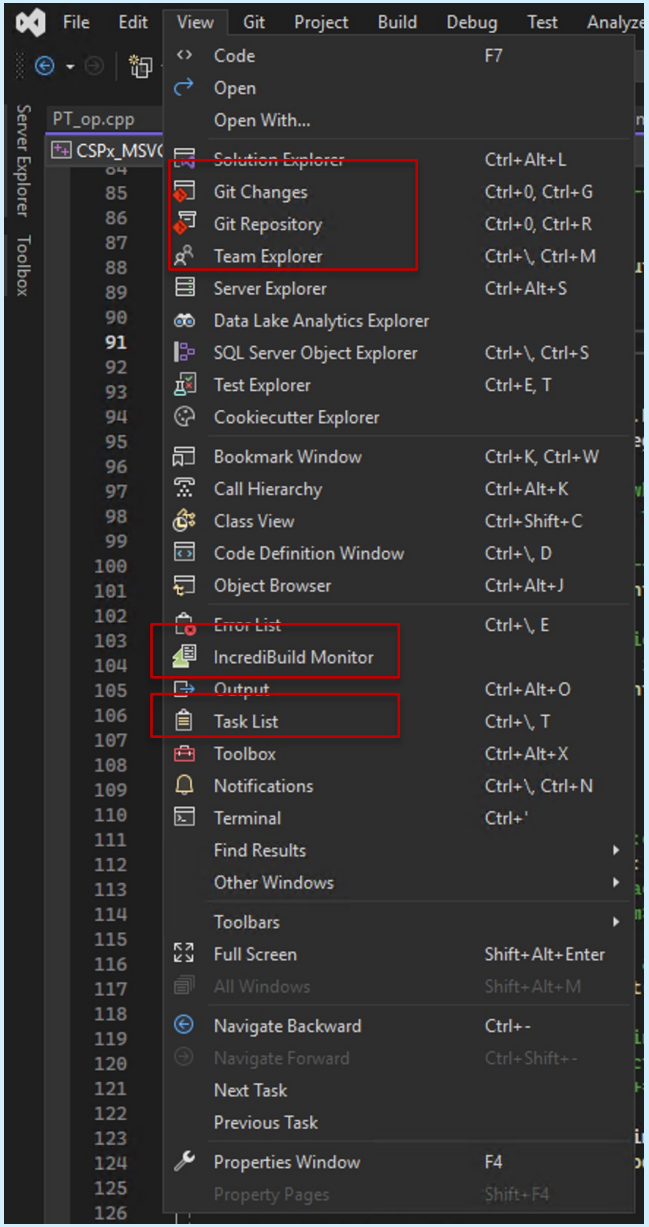


Imperial College London

MSVC Windows

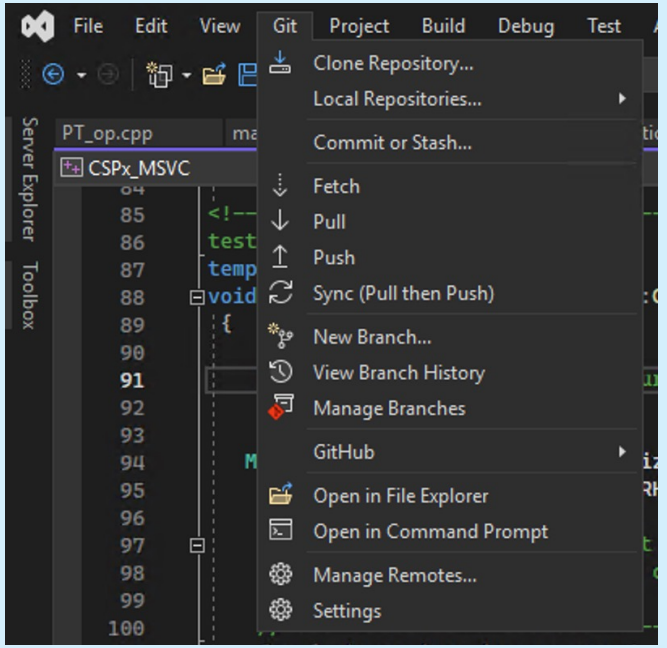
Monitoring

Useful windows to monitor coding/compiling coding



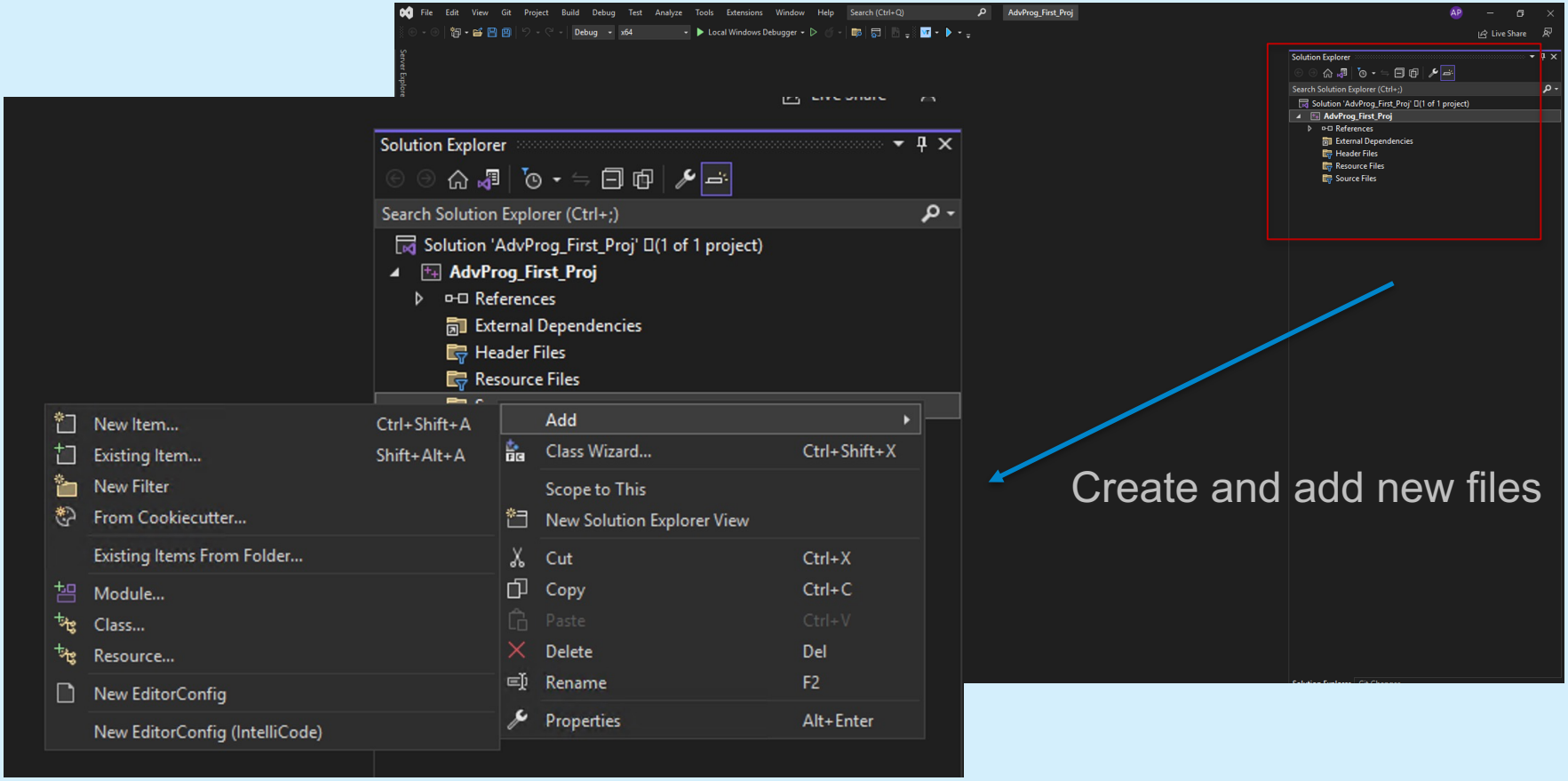
Git

You can connect directly to your git repository



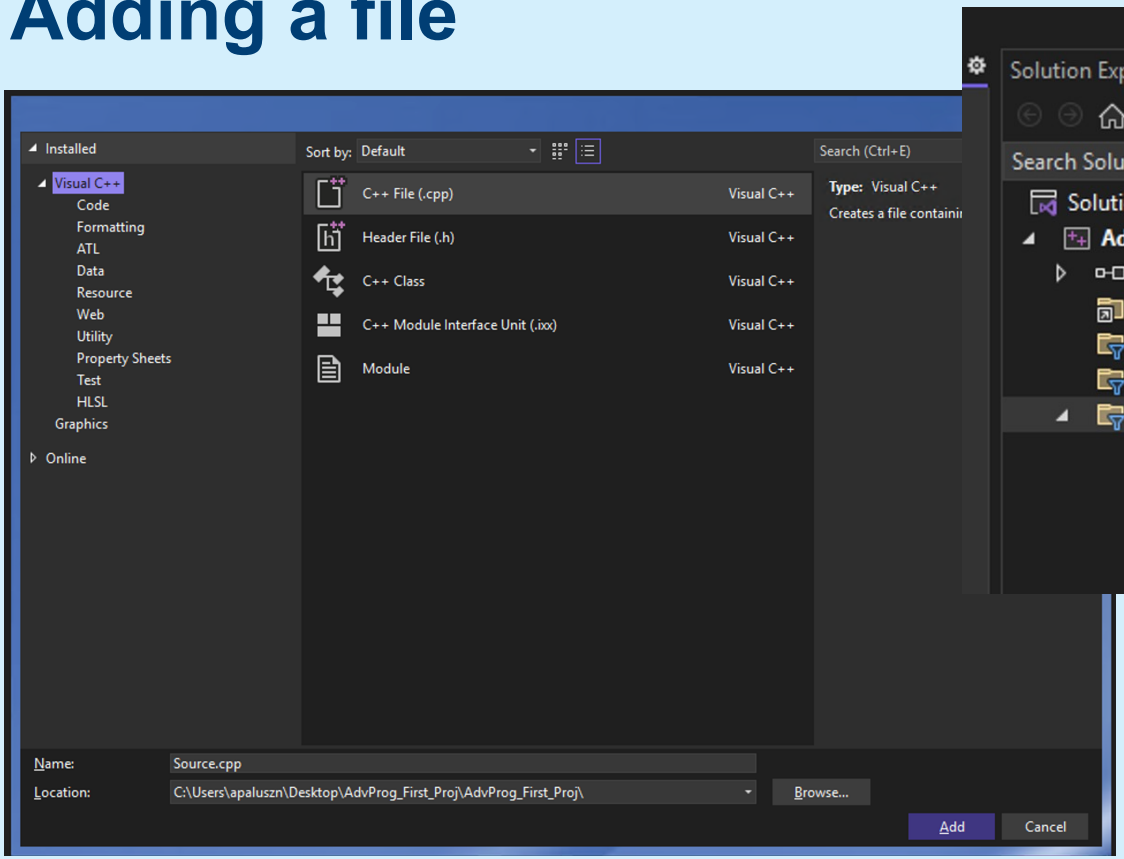
MSVC Windows

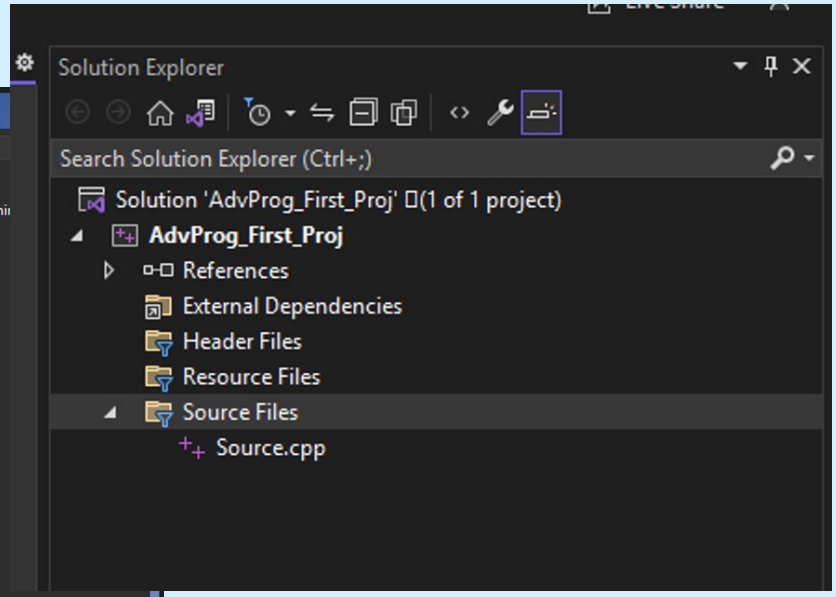
Visual Studio Community 2022: Source/Headers



MSVC Windows

Adding a file

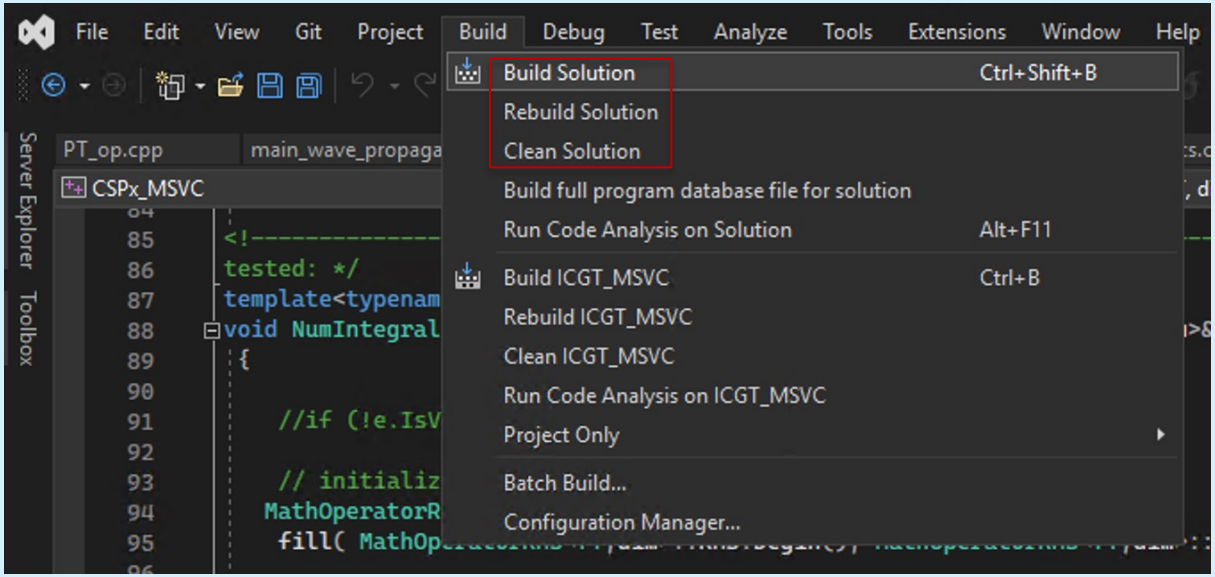




MSVC Windows

Visual Studio Community 2022: Compile

- Compile
- Clean
- Rebuild (Clean+Compile)



VS Code Mac

MAC Users

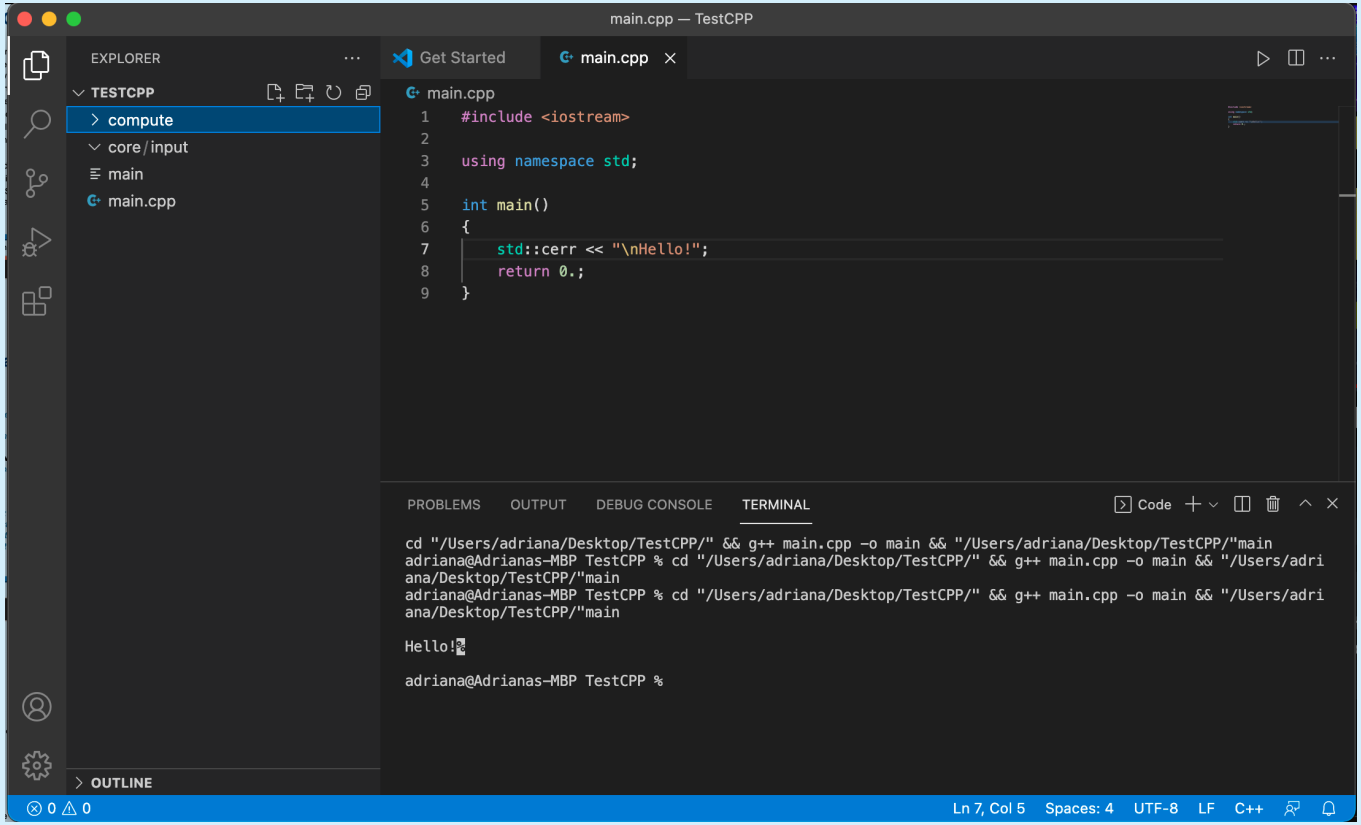
[similar setup, but more restricted functionality as compared to MSVC Community for Windows]

Visual Studio Code

Extension: Runner

Extension: C/C++

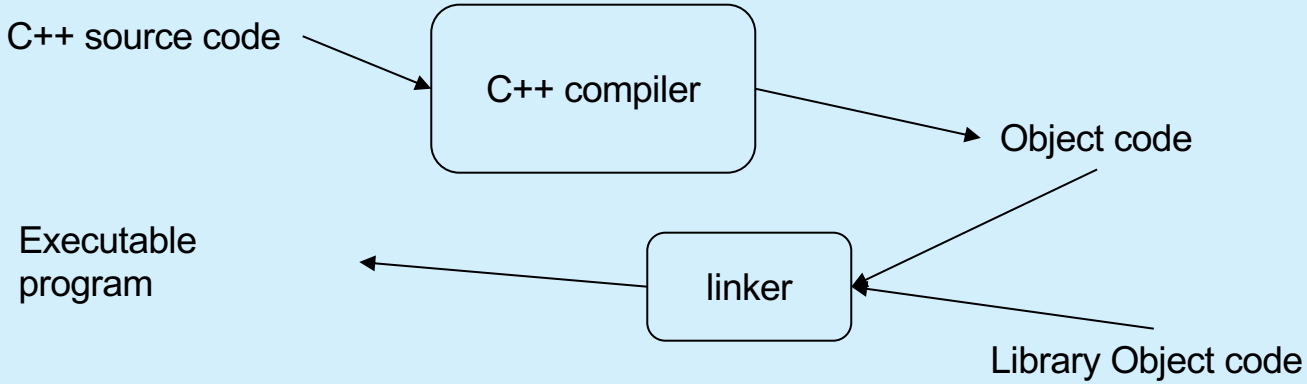
[lightweight] [editor]



A first program – hello.cpp

```
// a first program:

#include <iostream>   // get the library facilities needed for now

int main()            // main() is where a C++ program starts
{
    std::cout << "Hello, world!\n";   // output the 13 characters  Hello, world!
                                      // followed by a new line
    return 0;                         // return a value indicating success
}

    // note the semicolons; they terminate statements
    // braces { … } group statements into a block – the block defines the “scope” of a variable
    // main( ) is a function that usually takes no arguments ( )
    //      and returns an int (integer value) to indicate success or failure
```

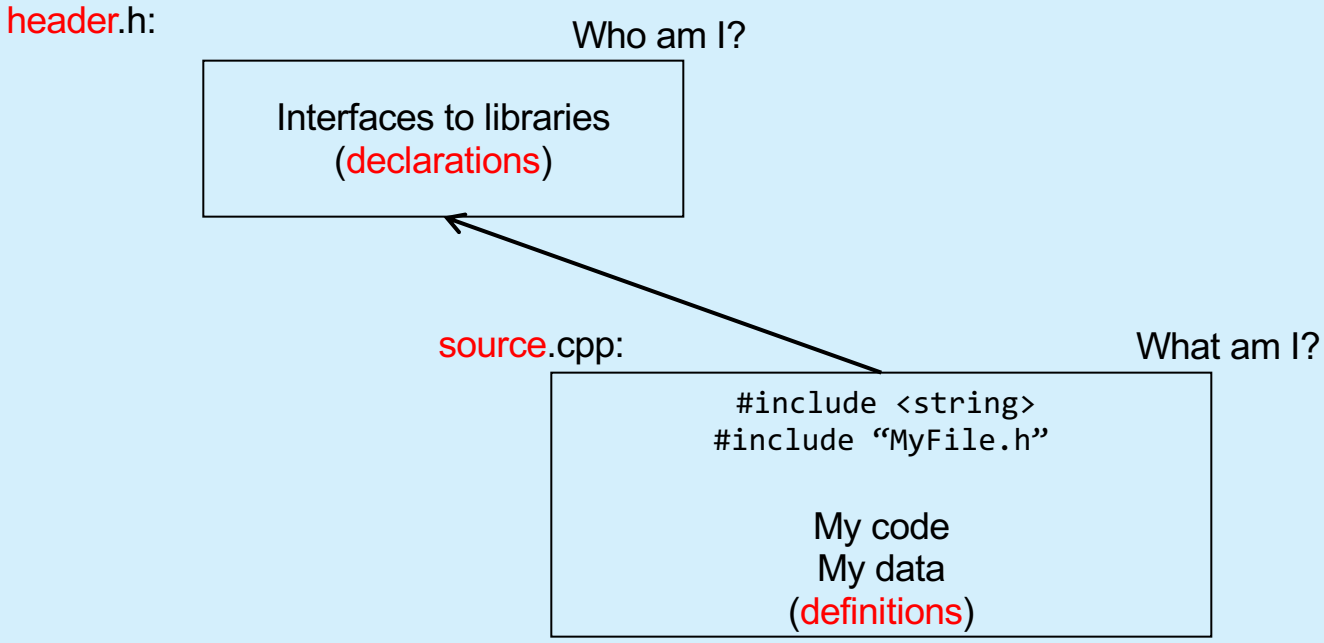
Compilation and linking



- You write C++ source code
  - Source code is (in principle) human readable
- The compiler translates what you wrote into object code (sometimes called machine code)
  - Object code is simple enough for a computer to "understand"
- The linker links your code to system code needed to execute
  - E.g., input/output libraries, operating system code, and windowing code
- The result is an executable program
  - E.g., a **.exe** file on windows or an **a.out** file on Unix

We will revisit this!

Source files



Integers and Strings

- Strings
  - `cin >>` reads a word
  - `cout <<` writes
  - `+` concatenates
  - `+= s` adds the string **s** at end
  - `++` is an error
  - `-` is an error

  …
- Integers and floating-point numbers
  - `cin >>` reads a number
  - `cout <<` writes
  - `+` adds
  - `+= n` increments by the int `n`
  - `++` increments by `1`
  - `-` subtracts

  …

The type of a variable determines which operations are valid and what their meanings are for that type

(that's called “overloading” or “operator overloading”)

Names

- A name in a C++ program
  - Starts with a letter, contains letters, digits, and underscores (only)
    - `x, number_of_elements, Fourier_transform, z2`
    - Not names:
      - `12x`
      - `time$to$market`
      - `main line`
    - Do not start names with underscores: `_foo`
      - those are reserved for implementation and systems entities
  - Users can't define names that are taken as keywords
    - E.g.:
      - `int`
      - `if`
      - `while`
      - `double`

Imperial College London

C++ Keywords

Standard C++ keywords

alignas
alignof
and [b]
and_eq [b]
asm [a]
auto
bitand [b]
bitor [b]
bool
break
case
catch
char
char8_t [c]
char16_t
char32_t
class
compl [b]
`concept` [c]
const
const_cast
`consteval` [c]
constexpr
`constinit` [c]
continue
`co_await` [c]
`co_return` [c]
`co_yield` [c]
decltype
default
delete
do
double
dynamic_cast
else
enum
explicit
`export` [c]
extern
false
float
for
friend
goto
if
inline
int
long
mutable
namespace
new
noexcept
not [b]
not_eq [b]
nullptr
operator
or [b]
or_eq [b]
private
protected
public
register
reinterpret_cast
`requires` [c]
return
short
signed
sizeof
static
static_assert
static_cast
struct
switch
template
this
thread_local
throw
true
try
typedef
typeid
typename
union
unsigned
using declaration
using directive
virtual
void
volatile
wchar_t
while
xor [b]
xor_eq [b]

Names

- Choose meaningful names
  - Abbreviations and acronyms can confuse people
    - `mvp, ufo, dh, mbf`
  - Short variable names can be meaningful
    - (only) when used conventionally:
      - **x** is a local variable
      - **i** is a loop index
  - Don't use overly long variable names
    - Ok:
      - `total_sum`
        `triangle_count`
        `temperature_average`
    - Too long:
      - `the_number_of_stars`
        `number_of_planets_in_the_galaxy`

Types and literals

- Built-in types
  - Boolean type
    - `bool`
  - Character types
    - `char`
  - Integer types
    - `int`
    - `short` and `long`
  - Floating-point types
    - `double`
    - `float`
- Standard-library types
  - `string`
  - `Fstream`
- And more…

- Boolean literals
  - `true false`
- Character literals
  - `'a', ‘z', ‘7', '\n', '$'`
- Integer literals
  - `0, 1, 42, 700, -6, 034, 0xa3`
- Floating point literals
  - `1.2, 17.345, .3, -0.54, 1.2e3, .3F`
- String literals ”`ASMR`", ”`Keep calm and carry on`"

If (and only if) you need more details, see the Stroustrup book

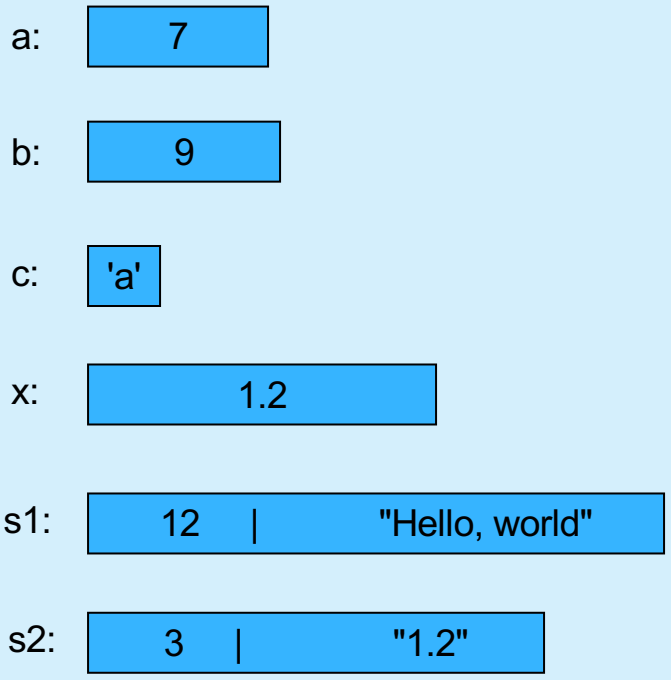
Types

- C++ provides a set of types
  - e.g. `bool, char, int, double`
  - Called “built-in types”
- C++ programmers can define new types
  - Called “user-defined types”
  - We'll get to that eventually
- The C++ standard library provides a set of types
  - e.g. `string, vector, complex`
  - Technically, these are user-defined types
    - they are built using only facilities available to every user

We will discuss memory management, and the memory footprint of these variables later in the course

Declaration and initialisation

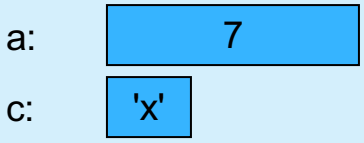
```
int a = 7;

int b = 9;

char c = 'a';

double x = 1.2;

string s1 = "Hello, world";

string s2 = "1.2";
```



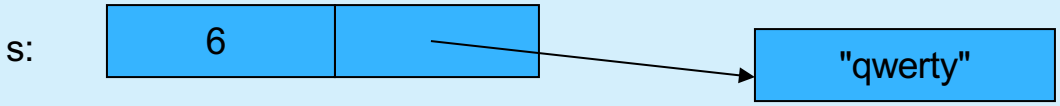
Objects

- An object is some memory that can hold a value of a given type
- A variable is a named object
- A declaration names an object

```
int a = 7;
char c = 'x';
```




```
string s = "qwerty";
```



Type safety

- Language rule: type safety
  - Every object will be used only according to its type
    - A variable will be used only after it has been initialized
    - Only operations defined for the variable's declared type will be applied
    - Every operation defined for a variable leaves the variable with a valid value
- Ideal: static type safety
  - A program that violates type safety will not compile
    - The compiler reports every violation (in an ideal system)
- Ideal: dynamic type safety
  - If you write a program that violates type safety it will be detected at run time
    - Some code (typically "the run-time system") detects every violation not found by the compiler (in an ideal system)

Quick guide to safety

Safe conversions

bool to **char**
bool to **int**
bool to **double**
char to **int**
char to **double**
int to **double**

Unsafe conversions

double to **int**
double to **char**
double to **bool**
int to **char**
int to **bool**
char to **bool**

A first program – hello.cpp

```
// a first program:

#include <iostream>   // get the library facilities needed for now

int main()            // main() is where a C++ program starts
{
    std::cout << "Hello, world!\n";   // output the 13 characters  Hello, world!
                                      // followed by a new line
    return 0;                         // return a value indicating success
}

    // note the semicolons; they terminate statements
    // braces { … } group statements into a block – the block defines the “scope” of a variable
    // main( ) is a function that usually takes no arguments ( )
    //      and returns an int (integer value) to indicate success or failure
```

Stream Basics

- Streams are a way of reading data from a device or sending it to a device
  - Later we will use streams to read and write data from files
- `cout` is the standard out, which is usually the screen, but can be redirected to other outputs, such a file or a printer
- `cin` is the standard in, which is usually the keyboard, but can be redirected from a file for instance
- `cerr` is the “error” stream, which is not buffered (cout are buffered)

Stream Basics

- You can send data to a stream using `<<` and receive data from a stream using `>>`
- cout can only receive data (i.e. only `<<` can be used with it) and `cin` can only send data (i.e. only `>>` can be used with it)
  - Later we will encounter file streams that can both send and receive data
- `endl` is used to indicate a new line. A new `cout` does not cause a new line in the output

Special Characters

- There are some special characters that can be used in strings (must be inside the string "..."):
- `\n`: New line (essentially the same as `endl`)
- `\t`: Tab space
- `\"`: Displays a " character
- `\'`: Displays a ' character
- `\\`: Displays a \ character

Comments

- In any programming it is useful to add comments to your code so that you (and other people) can know what you did and why
- You can either comment out sections using `/* ... */` where everything between `/*` and `*/` is ignored by the compiler
- ... or you can use `//` ,where compiler ignores the rest of the line that occurs after `//`

Some thoughts

- Programming involves tradeoffs.
- You must have ideals, but they often conflict, so you must decide what really matters for a given program.
  - Type safety
  - Run-time performance
  - Ability to run on a given platform
  - Ability to run on multiple platforms with same results
  - Compatibility with other code and systems
  - Ease of construction
  - Ease of maintenance
- Spend time on correctness or testing
- Aim for type safety and portability - always

Imperial College London

EXERCISE

- Write a program that, when run, greets the user by printing “Hello, world!” on the screen.
- If you complete the previous too quickly, try also adding the following functionality: After printing “Hello, world!”, the programme should print a message on the screen asking the user to enter their name. The program greets the user by name by printing the “Hello,” followed by the user’s name.
- If you have already installed MSVC or alike, you should create the code in your installed editor/IDE. If you haven’t installed an editor yet, try writing your code using an online editor, such as: https://cpp.sh

Imperial College London

Coursework 1

Imperial College London

Individual Coursework 1:

- You have all week to work on it
- Code that describes the scene
- 1-page description of code/design
- Class/es and main
- Written in C++!
- Due on Friday at 4pm

- Be creative!

Advanced Programming 2023/2024 – Coursework – Part A
Hand in by: Week 1, Friday 8th March @ 4 pm (max 25 marks)

Consider one of these pictures (A-I). Write one or more objects that describe the scene. Write a main that uses one or more of the objects in a creative manner to describe the scene in the picture. Submit your code via GitHub alongside a 1-page pdf with an explanation of the structure of your code (class/es and main).

Consider the interaction between multiple objects and consider creating different instances of the object(s) you create. Consider designing and implementing a few well-developed objects as opposed to many underdeveloped objects. Specify clearly in your main which object (or group of objects) you have chosen.


A


B


C


D


E


F


G


H


I