An Introduction to C++

A Brief History of C++

- In the late 1970's and early 1980's the C programming language was growing in popularity and usage in a variety of domains
  - It was general purpose, fast, portable and reasonably easy to use and build compilers/toolchains for
- While widely used, C did not encourage or enforce sound principles and practices for Software Engineering that were emerging
  - With languages like Simula and Smalltalk introducing objects, classes, and other language constructs capable of supporting large-scale software development, work began on creating an object-oriented version of C

A Brief History of C++

- From these efforts, two main approaches to an object-oriented C language appeared in the early 1980's:
  - C with Classes, created by Bjarne Stroustrup
  - Objective-C, created by Brad Cox and Tom Love
- In 1983, Rick Mascitti first referred to C with Classes tongue-in-cheek as C++, giving a nod to the ++ increment operator in C and the name stuck
- C++ generally proved more popular than Objective-C as the syntax seemed more natural to the average C programmer

A Brief History of C++

- Objective-C still did fairly well and had a role to play in the development of modern computing
  - It was adopted and promoted by NeXT and developed a reasonable following, but still didn't achieve significant traction
  - That changed when Apple acquired NeXT in 1996/1997 and used its technologies in creating OS X, thrusting Objective-C to the forefront for developing software for Apple's platforms
  - When Apple introduced Swift in 2014 (referred to by some as "Objective-C without the C"), this started a decline in the language's fortunes, however

A Brief History of C++

- C++ has evolved greatly over the years since its humble beginnings
  - Its first version included the basics, with classes, virtual functions, overloading, and improvements in typing and storage over C
  - In 1989, C++ 2.0 was released with multiple inheritance, abstract classes, and static members, with templates, namespaces and exceptions soon to follow
  - In 1998, C++ was standardized by the ISO as C++98 and a subsequent release in 2003 as C++03 without significant changes
  - C++11 in 2011 was the next major release expanding the standard library and adding features, with a revision in 2014 as C++14 and in late 2017 as C++17
  - The latest version is C++20 from late 2020, nearly missing the cut-off, and another revision is expected later in 2023

A Brief History of C++

- C++ has influenced the development of a number of other languages over the years as well
  - This include Lua, Perl, Python, PHP, Rust, and Go
  - Also notably includes Java, which was created as an alternative to C++ providing better portability and garbage collection
  - C++ also inspired the creation of C#, either directly or indirectly (depending on which side you believe in the "C# is a clone of Java" debate)
- Regardless, C++ has had wide influence over modern programming and software development

Why C++?

- C++ is the third most popular programming language according to the latest TIOBE index for 2023, behind Python and C (but ahead of Java once again)
- It is often praised for generally doing a good job of merging the benefits of object-orientation with the power and speed of C
- Its template library provides a decent collection of starting points, and add-on packages like Boost take things even further
- Mastering C++ will also make you a better programmer at C, Java, C# and many other languages – approaching things the other way is significantly more challenging

Why C++?


Always has been
Wait, it's all C and C++
libraries underneath
my favourite language?

Why Not C++?

objects could ask question about itself.
aka. allows type-identifying.
i.e. isInt(a)

- C++ lacks garbage collection and forces users to manage memory on their own (though some people see this as a plus)
- C++ lacks some high level language features like reflection
- Templates, while beneficial, are still harder to use and more error prone than they should be in practice
- Some recent developments in the language are perceived as feature creep and unnecessary

the object does not know "what am I"