## Context

The course will be using C++ 17.

Modern C++ is powerful and fast. It can be difficult, but it doesn’t have to be. Modern C++ is clean and simple, has differences to old C++, but the standard library does a lot of the work for us now.

ISO Standards committee official page: isocpp.org

Reference page: en.cppreference.com/w/

## Tools

**A compiler and a linker:**

The **Building** process is made up of compilation and linking. Source files **.cpp** are compiled to **.o/.obj** depending on the system. The object files will then be linked together to create the exe. Why are two separate processes required then? Because when singular files are changed in a massive system, this allows only the changed files to be recompiled and linked to the others, increasing build speed for each cycle. Imported libraries are also never changed, unless updated, so they will only be compiled once.

A context-aware text editor with documentation and intellisense. (Visual Studio Community 2022)

Debugger

**Popular tools:**

* GNU
* Compiler
  + Clang
  + GCC
* IDE: (Integrated Development Environment)
  + Visual Studio
    - Community edition – free for all purposes
    - Professional – for big companies
    - Enterprise – additional features
  + Visual Studio Code (lacks a built-in compiler and requires console building)
  + CLion
* Online compilers:
  + Great for comparing compilers
  + Testing new features on compilers, which support them
  + Sharing snippets

**Kinds of applications**

C++ is general-purpose, so it can be used for building programs on all kinds of platforms.

* Console applications – **Used for the course**
* PC apps – Windows/Linux/Mac
* Phone games and apps
* Etc…

**Making a project into Visual Studio**

When a new project is created depending on the type Visual Studio will include precompiled headers. Which are okay for large projects, but here are not needed.