① Prerequisite

→ Intermediate level of programming in any language

→ Ideally know how of how to execute a program ideally from a terminal.

→ Software debugging

→ Familirisation with oop (not necessary)

Courses

1) Foundation
2) Object oriented programming
3) Memory management
4) Concurrency

Projects

→ 5 different projects
- *) Route planner
- *) System monitor
- *) Garbage collector
- *) Traffic simulation
- *) Capstone project

② Introduction to C++

→ Compiled programming language
→ C++ 11 turns C++ into a totally modern programming language.

→ C++ used in embedded devices, IOT, automotive sectors

→ C++ 17 was a major release but C++ 11 was one of major release.

→ C++ 11 includes :-
- *) Move Sematics
- *) Variadic templates
- *) Initialiser lists
- *) auto keyword
- *) Lambda expression
- *) null pointer
- *) constant expression
- *) range-based for loops
- *) smart pointers.

→ C++ 17 includes :-
- *) Standard filesystem with STL
- *) Standard string View
- *) Parallel implementation of many STL
- *) Inline variables

Subject: / /

→ C++ 20 : Minor release that includes

*) Generic programming with templates

Standard Library

→ C++ standard library is a collection of classes and functions, which are written in the core language and part of the C++ ISO standard itself.

→ It is preferable to utilize functionality that already exists in the standard library, instead of implementing it from scratch.

→ C++ core guidelines (SL.1) states that :-

"Use libraries wherever possible"
Reason : Save time. Don't reinvent the wheel. Don't replicate the work of others.

and SL.2

"Prefer the standard library to other libraries"
Reason : More people know the standard library. It is more likely to be stable, well maintained, and widely available than your own code or most other libraries.

Subject: / /

Namespace

→ STL functions and classes exist in the std:: namespace.

Eg: std::vector

→ To use a STL feature, the corresponding header file should be included

Eg: #include <vector>

```cpp
#include <iostream>
#include <vector>


int main() {
    std::vector<int> intNum = {15, 5, 8};
    return 0;
}

```

Compilation

→ C++ is a compiled programming language, which means that programmers use a program to compile their human-readable source code into machine-readable object and executable files.

→ The program that performs this task is called compiler.

Subject: / /

→ C++ doesnot have any official compiler. Instead, there are many different compilers.

GNU compiler collection

→ It is a popular, open-source, cross platform compiler from the larger GNU project.

→ In particular, g++ is a command line executable that compiles C++ source code and automatically links the C++ standard library.

Linking

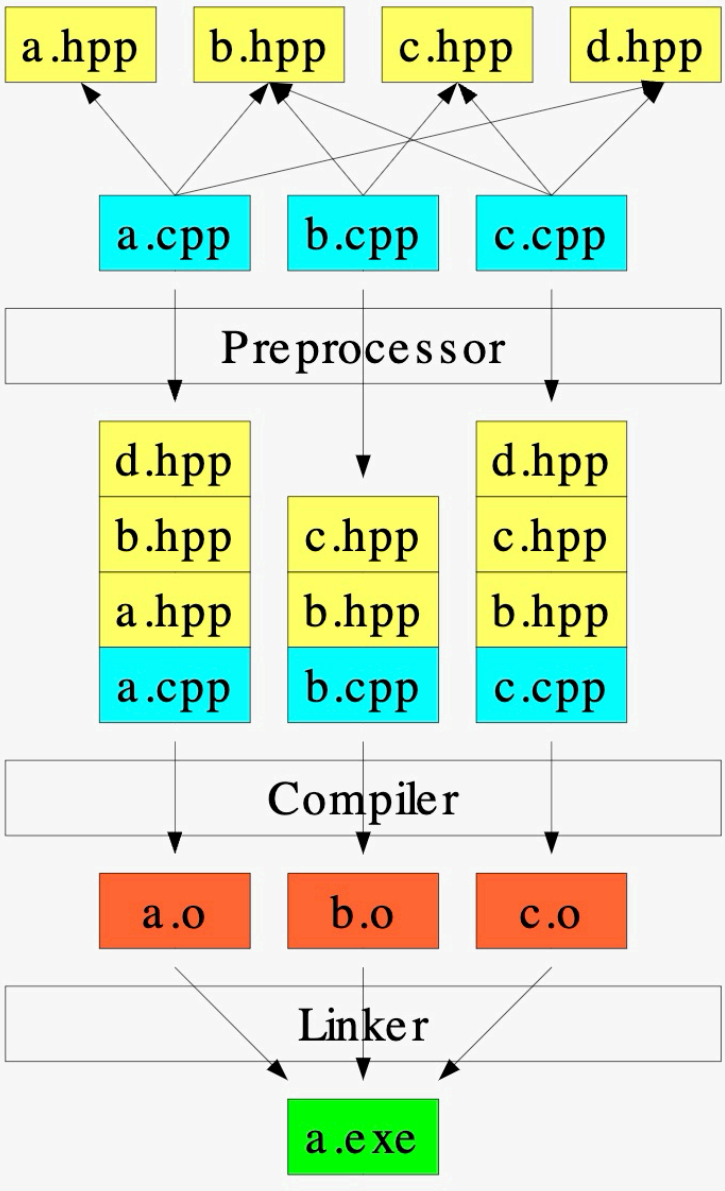
→ In order to use classes and functions from the C++ STL, the compiler must have access to a compiled version of the standard library, stored in object files.

→ Most compilers including gcc (GNU compiler collection) include those object files as part of the installation process.

→ In order to use the STL facilities, the compiler must link the standard library object files to the object files created from the programmer's source code.

Subject: / /

→ then once the linking is complete, it is able to generate a standalone executable.



C++ Compilation Process (Wikimedia)

Subject: / /

Build Tools

→ Make and CMake are two separate and similar build tools that both serve to help simplify the process of building software.

→ In particular, build tools automate the process of compiling multiple source code files into object files, linking those object files together and generating an executable.

→ Build tools also often automate the process of determining which files have changed since the last build and thus need to be recompiled.

Why use a build tool?

*) Managing complexity
*) Efficiency
*) Cross-platform compatibility
*) Automation

Compiled → translates source code into machine code, individually

Build Tools → crucial for managing builds efficiently and reliably.

Subject: / /

Make

→ GNU Make is a widely-used build tool that relies on Makefiles to automate the process of building a project.

→ A Makefile typically includes one or more targets. Each target performs a different actions.

→ build is a common target name that is configured in the Makefile to compile all of the project's source code into an executable file.

→ Clean, on the other hand, is a common target to delete all object files and other artifacts of the build process, resulting in a clean, unbuilt project state.

→ Running either Make build or Make clean on the command line cause Make to search for a local Makefile, search for a matching targets within that makefile, and then execute the target.

Subject: / /

CMake

→ CMake is a build tool that facilitates cross-platform builds, so that it is straightforward to build the same source code on Linux, macOS, Windows, or any other operating system.

→ CMake relies on CMakeLists.txt file can be a bit daunting

Installation

1) macos

in Terminal type Xcode-select --install

2) linux

```
Sudo apt update
Sudo apt install build-essential
Sudo apt install gdb
```

3) Windows
   *) MinGW
   *) Microsoft Visual Studio

```
g++ hello.cpp
```

Subject: Debugging / /

→ Debugging is an important part of Software development

→ Debuggers are tools that allow you to pause the execution of the code in various iorations, inspect the state of the program, and step through the code line by line.

→ GDB and LLDB are two popular, open Source debuggers for C++.