

Lesson 1 Teacher Lesson Plan

Level 1 C++ Foundations What Is C plus plus

Lesson 1 introduces students to programming in plain C++ before any ESP32 hardware work. The goal is not mastery. The goal is to see whether students can type carefully, run a program, make a small change, and recover from simple errors.

Lesson Snapshot

Item	Plan
Audience	Middle and high school robotics students
Length	60 to 75 minutes
Format	Teacher demo, guided practice, partner work, individual skill check
Materials	Windows laptop setup, VS Code, g++, projector, student workbook
Big idea	C++ source code is written by a human, translated by a compiler, then run by the computer.

Learning Objectives

- Explain that C++ is a programming language used to write instructions for computers and robot controllers.
- Define source code, compiler, program, compile, and run in beginner-friendly words.
- Identify the basic parts of a first C++ program: include line, main function, braces, cout, semicolons, comments, and return line.
- Edit a Hello World program and successfully compile and run it.
- Use a simple error message to find a missing semicolon or quotation mark.

Vocabulary

Term	Student Friendly Meaning
C++	A programming language used to write software.
Source code	The instructions a programmer writes.
Compiler	A tool that translates C++ into a program the computer can run.

main()	The starting point of this C++ program.
cout	A command that prints text to the terminal.
Syntax	The exact spelling and punctuation rules of a language.

Timing Guide

Time	Activity
0 to 5 min	Opening question: What counts as a set of instructions?
5 to 15 min	Mini lesson: source code, compiler, program, and the edit compile run cycle.
15 to 25 min	Teacher demo: create main.cpp, compile, and run Hello World.
25 to 40 min	Guided practice: students type and run the program.
40 to 55 min	Modify output lines and add comments.
55 to 70 min	Skill check and quick reflection.

Starter Program

```
#include <iostream>

using namespace std;

int main() {
    cout << "Hello, world!" << endl;
    return 0;
}
```

Teaching Notes

- Do not rush the first successful compile. For many students, this is the real win of Lesson 1.
- When explaining #include, say it lets the program use input and output tools. Save deeper detail for later.
- When explaining using namespace std, say it keeps today's cout line shorter. Save namespaces for later.
- Treat compiler errors as normal feedback, not failure.
- Model reading the first useful error line, then checking nearby code.

Skill Check Criteria

Skill	Evidence
Type code accurately	Program matches the starter program closely enough to compile.
Compile and run	Student can produce visible terminal output.
Make a small edit	Student changes printed text and recompiles.
Read simple structure	Student can point to main, braces, cout, and semicolons.
Debug one simple issue	Student fixes a missing semicolon, quote, or brace with support.

Exit Question

In one or two sentences, explain what the compiler does.