📘 C++ `struct` vs `class`

In C++, both `struct` and `class` are used to define **user-defined types** that can hold **data** and **functions**. While they are **very similar**, there are a few key differences that are important to understand.

🔹 What is a `struct`?

A `struct` (short for **structure**) is a way to group related variables (and optionally functions) under one name.

```
struct Point {
    double x;
    double y;

    void print() {
        std::cout << "(" << x << ", " << y << ")" << std::endl;
    }
};
```

✅ Characteristics of `struct`:

- **Default access modifier**: `public` 🔓
- Typically used for **plain-old-data (POD)** types.
- Supports **member functions**, **constructors**, **destructors**, **inheritance**, and **polymorphism** (just like classes).
- Often used in **C-style programming** or for **simple data containers**.

🔸 What is a `class`?

A `class` is a more formal and flexible way to define objects in C++. It encapsulates **data** and **behavior** and is the foundation of **object-oriented programming (OOP)** in C++.

```
class Circle {
private:
    double radius;

public:
    Circle(double r) : radius(r) {}

    double area() const {
        return 3.14159 * radius * radius;
    }
};
```

✅ Characteristics of `class`:

- **Default access modifier**: `private` 🔒
- Designed for **encapsulation**, **abstraction**, **inheritance**, and **polymorphism**.
- Encourages **data hiding** and **modular design**.
- Ideal for **complex systems** and **OOP design patterns**.

⚖️ Key Differences

Feature	`struct` 🧱	`class` 🏛️
Default access modifier	`public` 🔓	`private` 🔒
Inheritance default	`public`	`private`
Use case	Simple data containers	Full OOP design
Syntax	`struct Name { ... };`	`class Name { ... };`
Functionality	Same as class	Same as struct

🧠 When to Use What?

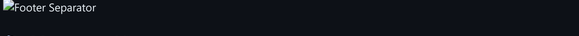
- Use `struct` when:
  - You need a **simple data structure**.
  - You want **public access** by default.
  - You're working with **interfacing C code**.
- Use `class` when:
  - You need **encapsulation** and **data hiding**.
  - You're building **complex systems**.
  - You want to leverage **OOP principles**.

🏏 Fun Fact

In C++, the only **technical difference** between `struct` and `class` is the **default access level**. Everything else—constructors, destructors, inheritance, etc.—works the same!

🎯 Quiz!

Here's a short quiz on the topic: quiz



🔭 Markdown Viewer

How to view the markdown files in a browser...

- Markdown Viewer

🧠 Lecture Practices

Here are the lecture Practices...

- Day 7
- Day 8
- Day 9

🔍 Lecture Quizzes

Here are the lecture quizzes...

- Day 7
- Day 8
- Day 9

Weekly Topics

Here are the topics for the week...

- Classes
- Structs
- Fields
- Getters and Setters
- Constructors
- Instances
- Inheritance
- Polymorphism
- Pointers
- Upcasting
- Misc. Concepts
- 4 Pillars of OOP