✅ What is an Abstract Class?

An **abstract class** in C++ is a class that **cannot be instantiated directly.** It serves as a **blueprint for derived classes**, and typically contains at least **one pure virtual function.**

A **pure virtual function** is declared using the syntax:

cpp Copy Edit

```cpp
virtual void functionName() = 0;
```

This tells the compiler that derived classes **must** override this function. Abstract classes are used when you want to define an **interface** or a **base behavior** but you don't want objects of the base class itself.

📜 Syntax:

cpp

Copy Edit

```cpp
class AbstractBase {
public:
    virtual void display() = 0;  // Pure virtual function
};
```

```cpp
#include <iostream>
using namespace std;

// Abstract class
class Shape {
public:
    virtual void area() = 0;  // Pure virtual function
};

// Derived class
class Circle : public Shape {
private:
    float radius;
public:
    Circle(float r) {
        radius = r;
    }
    void area() override {
        cout << "Area of Circle: " << 3.14 * radius * radius << endl;
    }
};
```

```cpp
int main() {
    // Shape s; // ❌ Error: cannot instantiate abstract class
    Circle c(5);
    c.area();  // ✅ Output: Area of Circle: 78.5
    return 0;
}
```

📝 Key Points:

- An abstract class **cannot be instantiated.**
- A class becomes abstract if it has **at least one pure virtual function.**
- Abstract classes are often used to define a **common interface** for a group of derived classes.
- Derived classes **must implement** all pure virtual functions, or they will also remain abstract.