# 📘 Upcasting in C++



## 🧠 What is Upcasting?

**Upcasting** is the process of converting a pointer or reference of a **derived class** to a **base class** type. It is called "upcasting" because in a class hierarchy, the base class is conceptually higher than the derived class.

```
class Animal {
public:
    virtual void speak() {
        std::cout << "Animal speaks\n";
    }
};

class Dog : public Animal {
public:
    void speak() override {
        std::cout << "Dog barks\n";
    }
};

Animal myPet = Dog();  // only copies the Animal parts of Dog to the myPet variable
Animal* aPtr = new Dog(); // 🔼 Upcasting. points aPtr to the Dog object
aPtr->speak(); // Output: Dog barks (due to polymorphism)
delete aPtr; // deallocate the Dog object
```

---

## ✅ Key Characteristics

- **Safe and implicit**: No cast operator is needed.
- **Supports polymorphism**: If the base class has virtual functions, the derived class's overridden methods will be called.
- **Loses access to derived-specific members**: Only base class members are accessible through the base class pointer/reference.

---

## 🧪 Why Use Upcasting?

### 🔁 Polymorphism

Allows writing flexible and reusable code that works with base class types but behaves according to the derived class implementation.

```
void makeItSpeak(Animal* a) {
    a->speak(); // Calls the appropriate speak() method
}
```

### 🧰 Code Reusability

You can store different derived objects in a single container of base class pointers:

```
std::vector<Animal*> zoo;
zoo.push_back(new Dog());
zoo.push_back(new Cat());
```

---

## ⚠️ Limitations

- You **cannot access** members that are only in the derived class.
- If you need to access derived-specific functionality, you must **downcast** (with care).

---

## 🧭 Summary

| Feature | Upcasting |
| --- | --- |
| Direction | Derived → Base |
| Safe? | ✅ Yes |
| Implicit? | ✅ Yes |
| Polymorphic behavior | ✅ Yes (if virtual functions are used) |
| Access to derived members | ❌ No |



## 🔭 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