

Understanding Inheritance and Base Class Constructors in C++

A Comprehensive Guide with Examples

Mohamed KHABOU

February 15, 2025

Objective

Objective

- ▶ Understand the concept of inheritance in C++.
- ▶ Learn how to properly call base class constructors.
- ▶ Explore common pitfalls and best practices.

What is Inheritance?

What is Inheritance?

- ▶ Inheritance allows a class (derived) to reuse properties and methods of another class (base).
- ▶ Example:
 - A Dog class can inherit from an Animal class.
 - The Dog class automatically gets all the methods and properties of Animal.

Inheritance Example

Inheritance Example

```
#include <iostream>

// Base class
class Animal {
public:
    void speak() {
        std::cout << "Animal speaks!" << std::endl;
    }
};

// Derived class
class Dog : public Animal {
public:
    void bark() {
        std::cout << "Dog barks!" << std::endl;
    }
};

int main() {
    Dog myDog;
    myDog.speak(); // Inherited from Animal
    myDog.bark();  // Defined in Dog
    return 0;
}
```

Calling Base Class Constructors

Calling Base Class Constructors

- ▶ If the base class has a constructor with parameters, the derived class must explicitly call it.
- ▶ Example:
 - If `Animal` has a constructor `Animal(std::string name)`, the `Dog` constructor must call it.

Base Constructor Example

Base Constructor Example

```
#include <iostream>

// Base class
class Animal {
private:
    std::string name;
public:
    Animal(std::string n) : name(n) {
        std::cout << "Animal constructor: " << name << std::endl;
    }
    void speak() {
        std::cout << name << " speaks!" << std::endl;
    }
};

// Derived class
class Dog : public Animal {
public:
    Dog(std::string n) : Animal(n) { // Calling the base constructor
        std::cout << "Dog constructor: " << n << std::endl;
    }
    void bark() {
        std::cout << "Dog barks!" << std::endl;
    }
};

int main() {
    Dog myDog("Buddy");
    myDog.speak(); // Inherited from Animal
    myDog.bark();  // Defined in Dog
}
```

Common Pitfalls

Common Pitfalls

- Forgetting to call the base class constructor when it requires parameters.

```
class Animal {  
public:  
    Animal(std::string n) {}  
};  
  
class Dog : public Animal {  
public:  
    Dog(std::string n) { // Error: Forgot to call Animal(n)  
    }  
};
```

```
class Dog : public Animal {  
public:  
    Dog(std::string n) : Animal(n) { // Correct  
    }  
};
```

Comparison

Inheritance vs. Base Constructor Call

Inheritance

- ▶ **Purpose:** Reuse code from a base class.
- ▶ **Syntax:** `class Derived : public Base {}.`
- ▶ **Example:** `Dog : public Animal.`

Base Constructor Call

- ▶ **Purpose:** Initialize the base class properly.
- ▶ **Syntax:** `Derived(args) : Base(args) {}.`
- ▶ **Example:** `Dog(std::string n) : Animal(n) {}.`

Key Takeaways

Key Takeaways

- ▶ Inheritance allows a derived class to reuse base class properties and methods.
- ▶ Always call the base class constructor explicitly if it requires parameters.
- ▶ Use proper syntax: `Derived(args) : Base(args) .`