Tutorial 44 - Virtual Base Class in C++

Concept Overview

A **virtual base class** prevents ambiguity in multiple inheritance by ensuring only one copy of the base class is inherited when multiple derived classes inherit from the same base class.

Example Scenario

1. Class `A` is the parent class of `B` and `C`.
2. Class `D` is derived from both `B` and `C`.
3. Without virtual inheritance, `D` would inherit two copies of `A`—one from `B` and another from `C`, causing **ambiguity** when accessing `A`'s members.

Solution:

- Use the `virtual` keyword when inheriting from `A` in `B` and `C`.
- This ensures `A`'s members are inherited only once, shared between all derived classes.

Syntax for Virtual Base Class

```
#include <iostream>
using namespace std;
class A {
public:
    void say() {
        cout << "Hello world" << endl;
    }
};
class B : public virtual A {
    // Virtual inheritance ensures a single copy of A's members
};
class C : public virtual A {
    // Virtual inheritance ensures a single copy of A's members
};
class D : public B, public C {
    // D now inherits only one copy of A's members
};
int main() {
    D obj;
    obj.say(); // No ambiguity, as A is a virtual base class
    return 0;
}

```

Explanation

1. **Class `A`**:
   - Contains a public method `say()` that prints "Hello world."
2. **Class `B` and `C`**:
   - Inherit `A` using the `virtual` keyword to ensure virtual inheritance.
3. **Class `D`**:

- Inherits from `B` and `C`. Only one copy of `A` is inherited, avoiding ambiguity.

4. **Main Function**:
   - An object of `D` calls the `say()` method without ambiguity.

Key Points

1. **Virtual Base Class**:
   - Use `virtual` keyword to ensure only one copy of the base class is inherited.
2. **Avoiding Ambiguity**:
   - Ensures no conflicts when accessing members of the base class.
3. **Inheritance Hierarchy**:
   - All derived classes share a single instance of the virtual base class.
4. **Syntax**:
   - `class Derived : public virtual Base { };`

Short Notes for Notebook

1. **Virtual Base Class**:
   - Prevents ambiguity in multiple inheritance.
   - Ensures only one copy of the base class is inherited.
2. **Use Case**:
   - When a class is indirectly inherited multiple times through different derived classes.
3. **Syntax**:

```
class B : public virtual A { };
class C : public virtual A { };
class D : public B, public C { };

```

4. **Key Benefits**:
   - Resolves ambiguity.
   - Shares a single instance of the base class.
5. **Example**:

```
obj.say(); // Accesses A's method without ambiguity
```