Tutorial 45 - Virtual Base Class in C++

Concept Overview

- A **virtual base class** resolves ambiguity in multiple inheritance by ensuring only one instance of the base class is inherited, regardless of the number of inheritance paths.

Example Scenario

- **Class** `Student`: Base class.
- **Classes** `Test` **and** `Sports`: Derived from `Student`.
- **Class** `Result`: Derived from both `Test` and `Sports`.

Problem:

- Without virtual inheritance, `Result` will inherit two copies of `Student`'s members (via `Test` and `Sports`), causing **ambiguity**.
- **Solution**: Use the `virtual` keyword to make `Student` a virtual base class.

Code Example

Program Code:

```
#include <iostream>
using namespace std;
class Student {
protected:
    int roll_no;
public:
    void set_number(int a) {
        roll_no = a;
    }
    void print_number() {
        cout << "Your roll no is " << roll_no << endl;
    }
};
class Test : virtual public Student {
protected:
    float maths, physics;
public:
    void set_marks(float m1, float m2) {
        maths = m1;
        physics = m2;
    }
    void print_marks() {
        cout << "Your result is here:" << endl
             << "Maths: " << maths << endl
             << "Physics: " << physics << endl;
    }
};
class Sports : virtual public Student {
protected:
    float score;
```

```cpp
public:
    void set_score(float sc) {
        score = sc;
    }
    void print_score() {
        cout << "Your PT score is " << score << endl;
    }
};
class Result : public Test, public Sports {
private:
    float total;
public:
    void display() {
        total = maths + physics + score;
        print_number();
        print_marks();
        print_score();
        cout << "Your total score is: " << total << endl;
    }
};
int main() {
    Result harry;
    harry.set_number(4200);
    harry.set_marks(78.9, 99.5);
    harry.set_score(9);
    harry.display();
    return 0;
}

```

Key Points

1. **Virtual Inheritance**:
    - Use `virtual` keyword while inheriting the base class:

    ```cpp
    class Test : virtual public Student { };
    class Sports : virtual public Student { };

    ```

2. **Avoids Ambiguity**:
    - Ensures `Student`'s members are inherited only once in `Result`.
3. **Protected Members**:
    - `roll_no`, `maths`, `physics`, and `score` are protected, so they are accessible in derived classes.

Execution Steps:

1. **Object Creation**:

    ```cpp
    Result harry;

    ```

2. **Set Data**:
    - Call `set_number()`, `set_marks()`, and `set_score()` to set roll number, marks, and score.
3. **Display Results**:
    - Call `display()` to print roll number, marks, score, and total.

Short Notes for Notebook

1. **Virtual Base Class**:
   - Prevents ambiguity in multiple inheritance.
   - Ensures only one copy of the base class is inherited.
2. **Syntax**:

```
class Derived : virtual public Base { };

```

3. **Key Features**:
   - Shared single instance of the base class across inheritance paths.
   - Avoids multiple copies of the base class in derived classes.
4. **Example Flow**:
   - Class `Student` (virtual base class).
   - Classes `Test` and `Sports` inherit `Student` virtually.
   - Class `Result` inherits `Test` and `Sports`.
5. **Main Code Flow**:

```
Result obj;
obj.set_number(4200);
obj.set_marks(78.9, 99.5);
obj.set_score(9);
obj.display();

```

6. **Output**:

```
Your roll no is 4200
Your result is here:
Maths: 78.9
Physics: 99.5
Your PT score is 9
Your total score is: 187.4
```