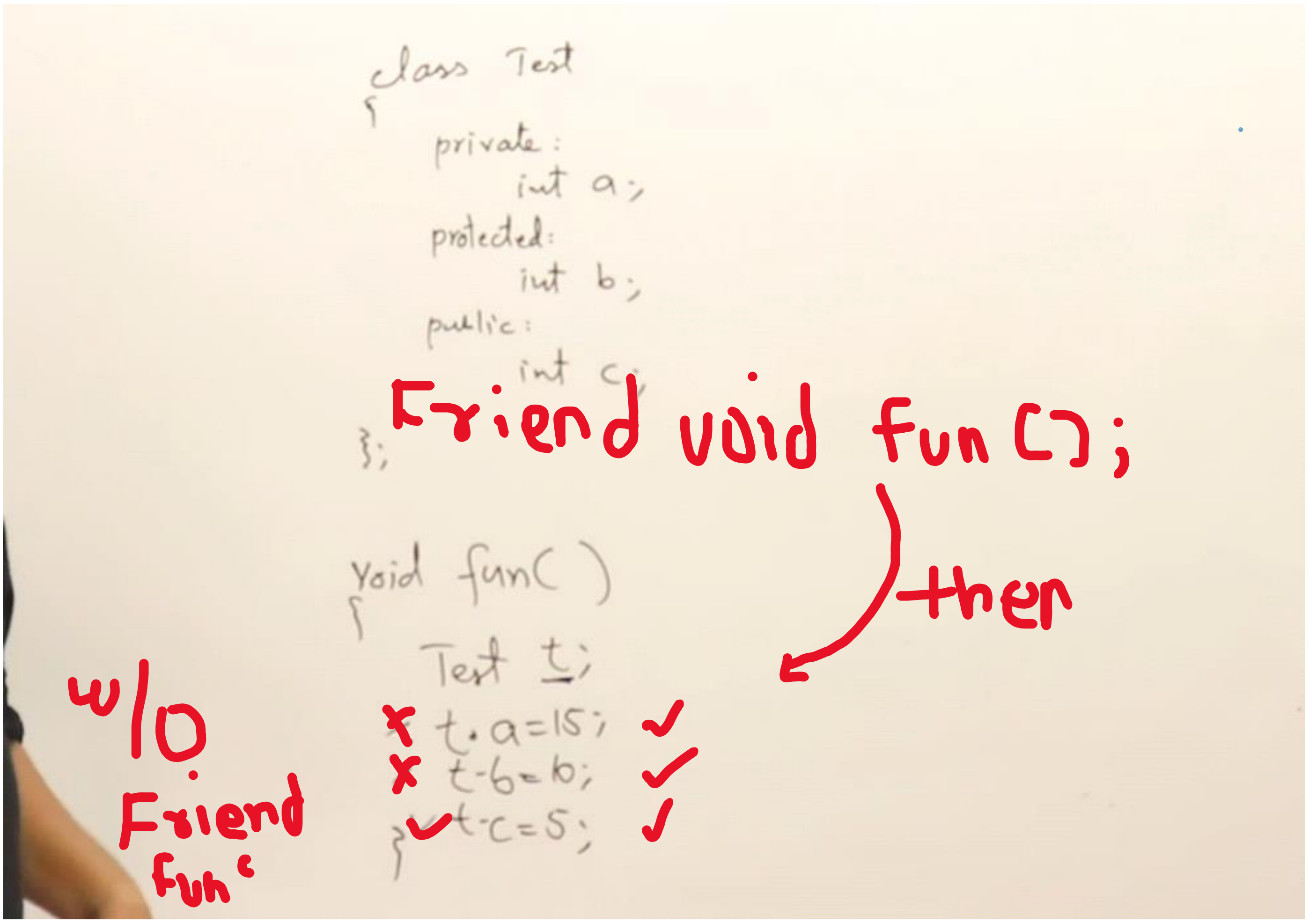
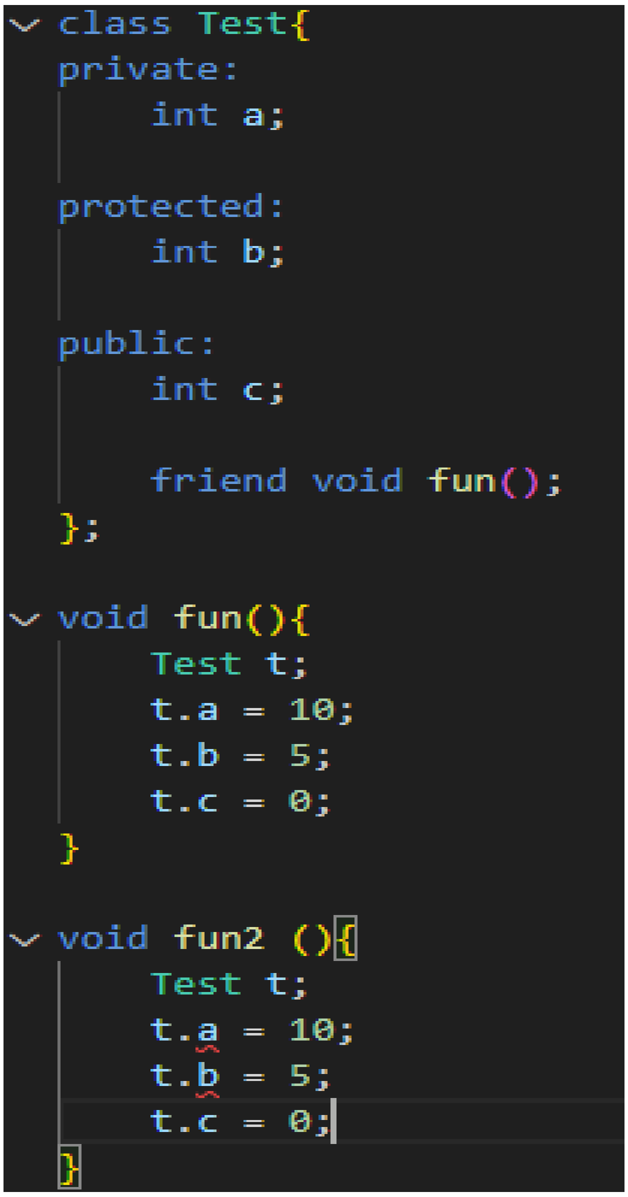
Friend and Static Members Inner Class

23 September 2024 11:03

In cpp a function outside the class is not allowed its private and protected members, hence they require a **friend function** inside the class.

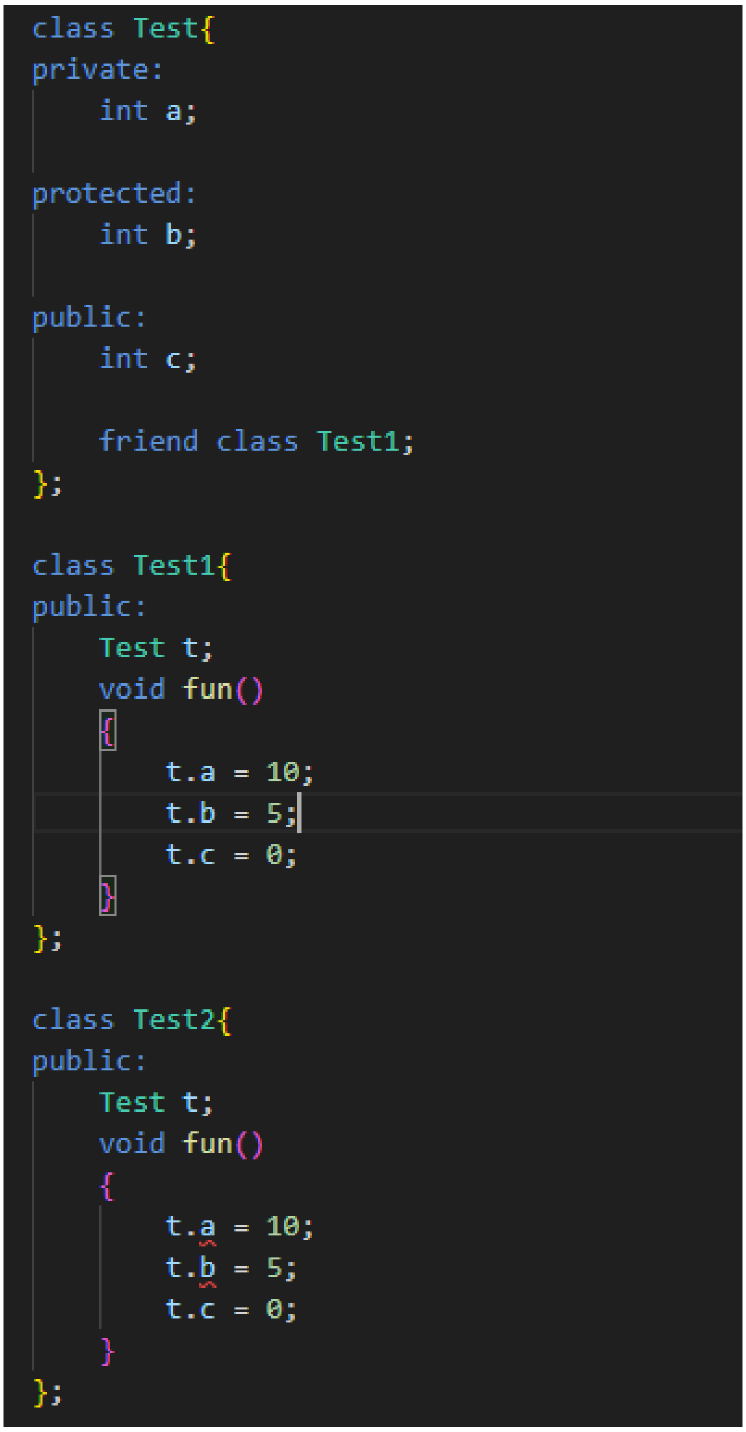


```
class Test
{
    private:
        int a;
    protected:
        int b;
    public:
        int c;
        Friend void fun();
};

void fun()
{
    Test t;
  ✗ t.a=15;  ✓
  ✗ t.b=6;   ✓
  ✓ t.c=5;   ✓
}
```



```cpp
class Test{
private:
    int a;

protected:
    int b;

public:
    int c;

    friend void fun();
};

void fun(){
    Test t;
    t.a = 10;
    t.b = 5;
    t.c = 0;
}

void fun2 (){
    Test t;
    t.a = 10;
    t.b = 5;
    t.c = 0;
}
```

Similarly Friend classes, another class cannot utilize members of another classes without inheriting. Hence, we require a **friend classes** to do the same.



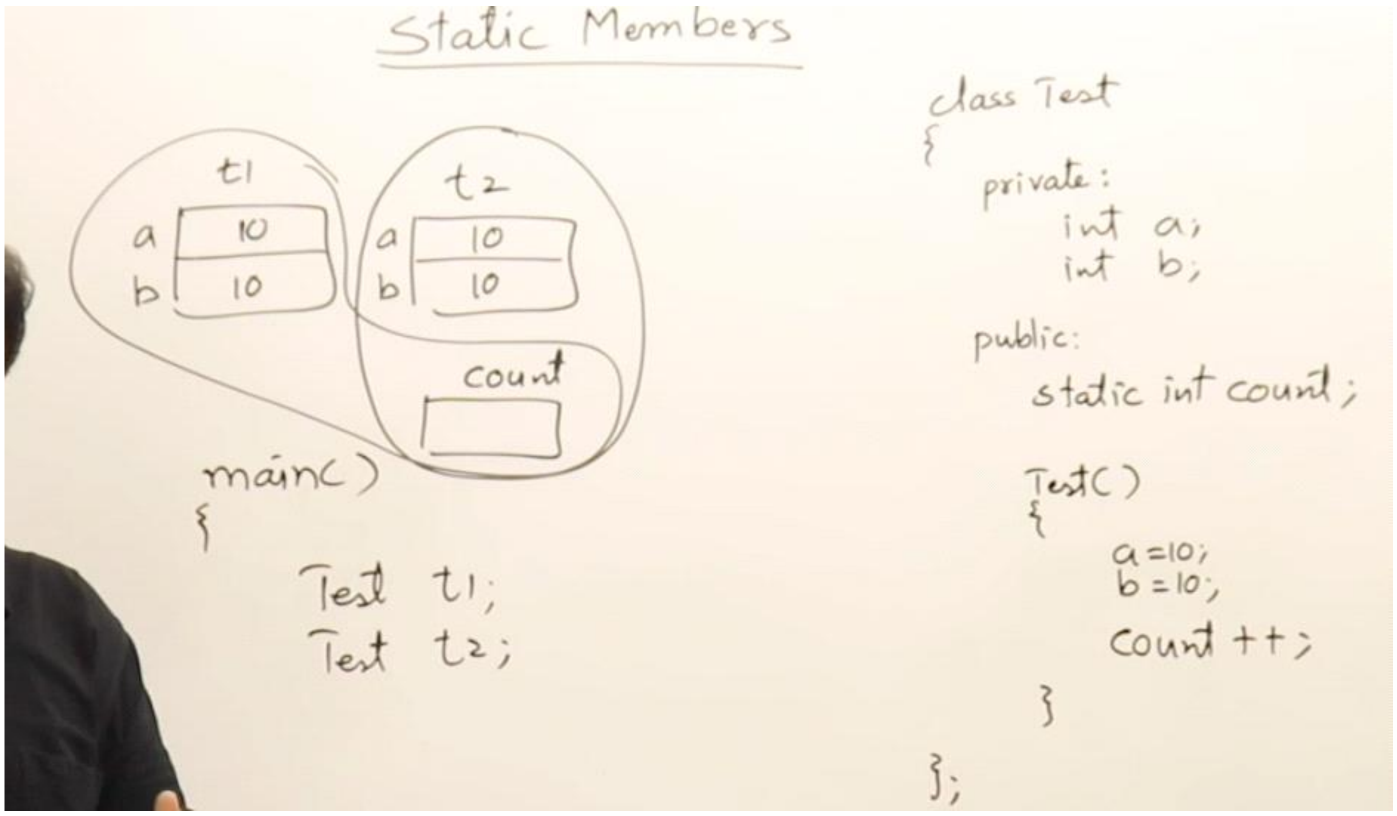
```cpp
class Test{
private:
    int a;

protected:
    int b;

public:
    int c;

    friend class Test1;
};

class Test1{
public:
    Test t;
    void fun()
    {
        t.a = 10;
        t.b = 5;
        t.c = 0;
    }
};

class Test2{
public:
    Test t;
    void fun()
    {
        t.a = 10;
        t.b = 5;
        t.c = 0;
    }
};
```

Static Member of a class

The static members belong to a class not to an object. Hence, every object share the same static memory location for static variable/ function/ member.

Static Members

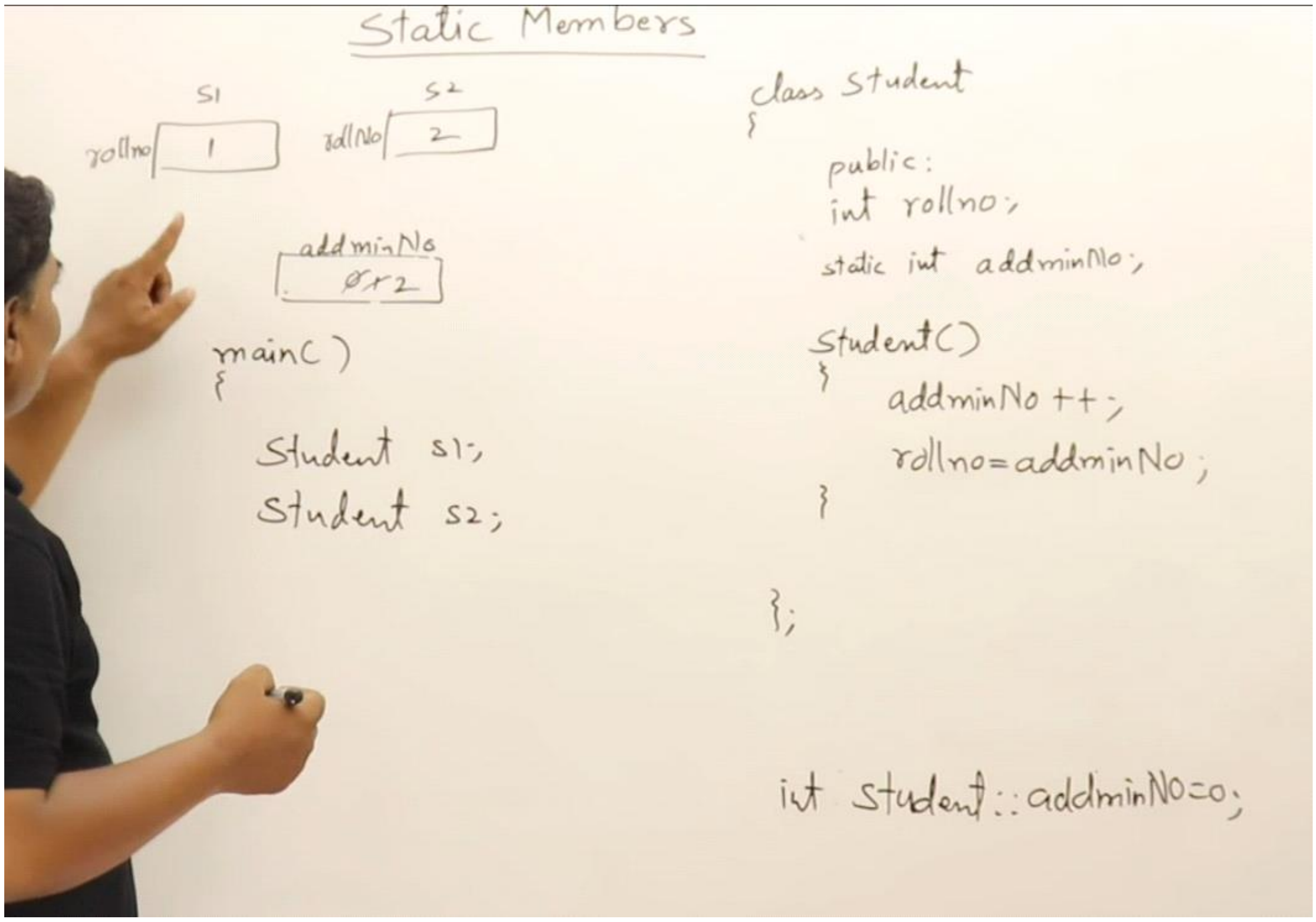
- Static data members are members of a class
- Only one instance of static members is created and shared by all objects
- They can be accessed directly using class name
-
- Static members functions are functions of a class, they can be called using class name, without creating object of a class.
- They can access only static data members of a class, they cannot access non-static members of a class.



```
class Test
{
  private:
      int a;
      int b;

  public:
     static int count;

     Test()
     {
        a=10;
        b=10;
        count++;
     }

};
```

```
main()
{
    Test t1;
    Test t2;
```

They are sharable members of a class.



```
class Student
{
   public:
   int rollno;
   static int addminNo;

   Student()
   {
      addminNo++;
      rollno=addminNo;
   }

};

int student::addminNo=0;
```

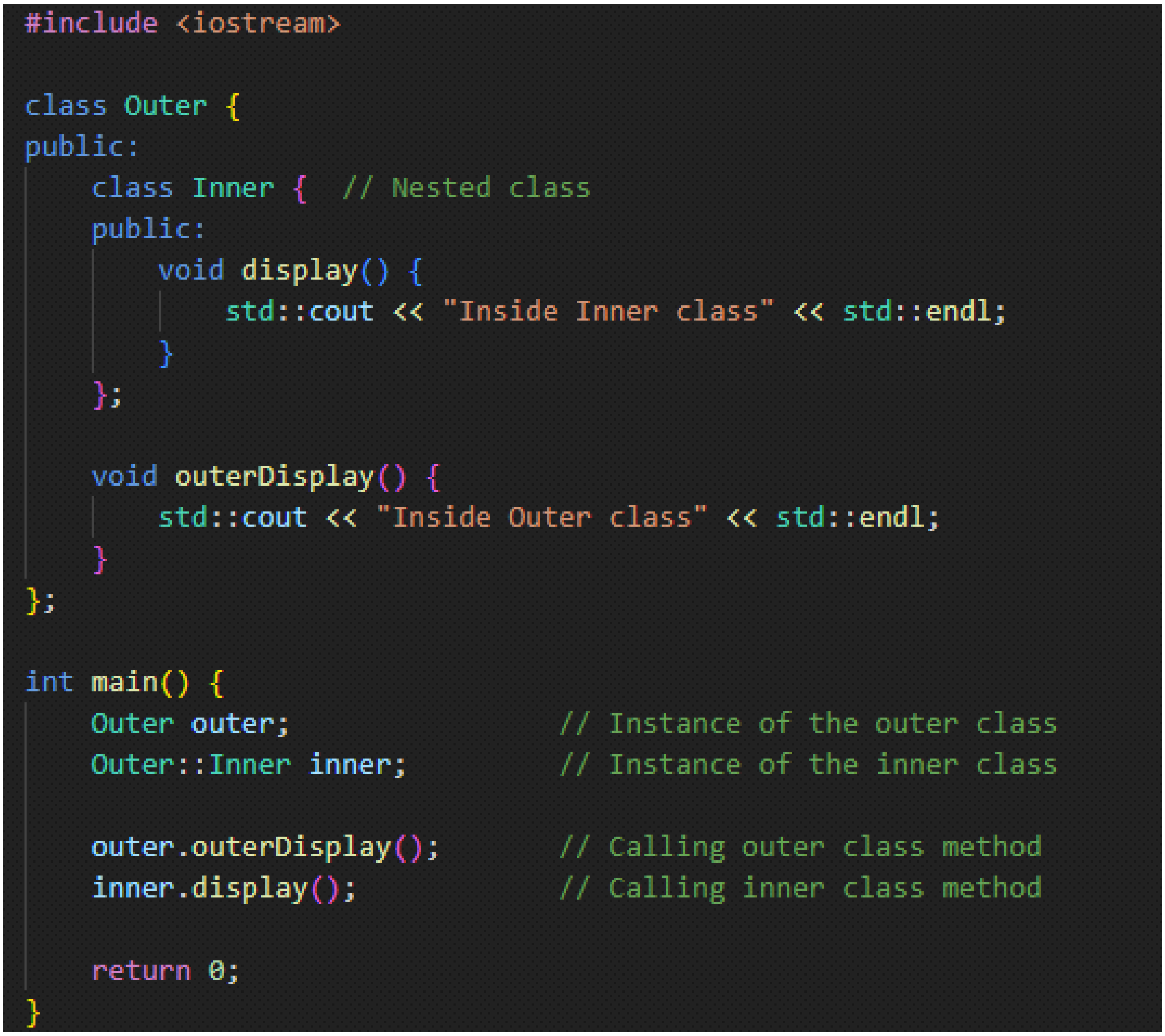
```
main()
{
   Student s1;
   Student s2;
```

Example :

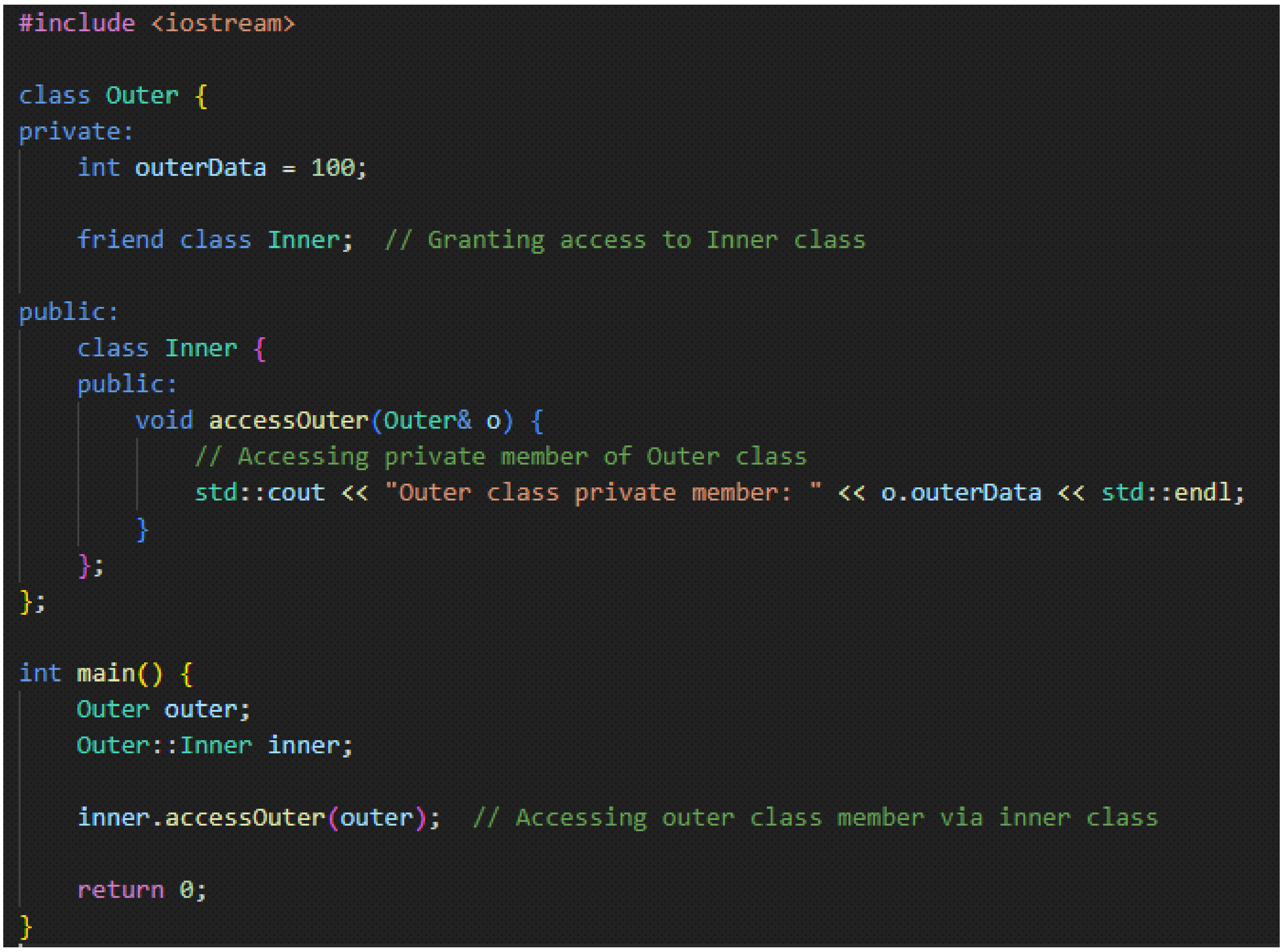
```
#include <iostream>
using namespace std;
class Student{
public:
    int roll;
    string name;
    static int addNo;
    Student(string n)
    {
        addNo++;
        roll = addNo;
        name = n;
    }
    void display()
    {
        cout << "Name " << name << endl
             << "Roll " << roll << endl;
    }
};
int Student::addNo = 0;
int main(){
    Student s1("John");
    Student s2("Ravi");
    Student s3("Khan");
    Student s4("Khan");
    Student s5("Khan");
    Student s6("Khan");
    s1.display();
    s6.display();
    cout << "Number Admission " << Student::addNo << endl;
}
```

Inner Class

an **inner class** (or nested class) is a class declared inside another class.
Nested classes can also access the enclosing class’s members (private or public) if declared as friend. But without friend they cannot.



```cpp
#include <iostream>

class Outer {
public:
    class Inner {  // Nested class
    public:
        void display() {
            std::cout << "Inside Inner class" << std::endl;
        }
    };

    void outerDisplay() {
        std::cout << "Inside Outer class" << std::endl;
    }
};

int main() {
    Outer outer;                    // Instance of the outer class
    Outer::Inner inner;             // Instance of the inner class

    outer.outerDisplay();           // Calling outer class method
    inner.display();                // Calling inner class method

    return 0;
}
```



```cpp
#include <iostream>

class Outer {
private:
    int outerData = 100;

    friend class Inner;  // Granting access to Inner class

public:
    class Inner {
    public:
        void accessOuter(Outer& o) {
            // Accessing private member of Outer class
            std::cout << "Outer class private member: " << o.outerData << std::endl;
        }
    };
};

int main() {
    Outer outer;
    Outer::Inner inner;

    inner.accessOuter(outer);  // Accessing outer class member via inner class

    return 0;
}
```

Summary of Nested Classes:

1. **Inner (Nested) classes** are declared inside another class.
2. They are useful for logically grouping classes and encapsulating related logic.
3. Inner classes can be declared `public`, `private`, or `protected` based on access requirements.
4. Inner classes can access private members of the outer class only if explicitly allowed (e.g., via `friend` declaration).
5. An instance of an inner class is typically declared as `Outer::Inner inner;`.