



## Module M22

Partha Pratim  
Das

Objectives &  
Outlines

Inheritance in  
C++

Data Members

Object Layout

Member  
Functions

Overrides and  
Overloads

Comparison

Module Summary

# Programming in Modern C++

Module M22: Inheritance: Part 2: Data Member & Member Function: Override & Overload

Partha Pratim Das

Department of Computer Science and Engineering  
Indian Institute of Technology, Kharagpur

*ppd@cse.iitkgp.ac.in*

*All url's in this module have been accessed in September, 2021 and found to be functional*



# Module Recap

## Module M22

Partha Pratim  
Das

Objectives &  
Outlines

Inheritance in  
C++

Data Members

Object Layout

Member  
Functions

Overrides and  
Overloads

Comparison

Module Summary

- Understood Hierarchy or ISA Relationship in OOAD
- Introduced the Semantics of Inheritance in C++

NPTEL



# Module Objectives

## Module M22

Partha Pratim  
Das

### Objectives & Outlines

Inheritance in  
C++

Data Members

Object Layout

Member  
Functions

Overrides and  
Overloads

Comparison

Module Summary

- Understand how inheritance impacts data members and member functions
- Introduce overriding of member function and its interactions with overloading

NPTEL



# Module Outline

## Module M22

Partha Pratim  
Das

### Objectives & Outlines

Inheritance in  
C++

Data Members

Object Layout

Member  
Functions

Overrides and  
Overloads

Comparison

Module Summary

## 1 Inheritance in C++

## 2 Data Members

- Object Layout

## 3 Member Functions

- Overrides and Overloads

## 4 Comparison

## 5 Module Summary



# Inheritance in C++

## Module M22

Partha Pratim  
Das

Objectives &  
Outlines

Inheritance in  
C++

Data Members

Object Layout

Member  
Functions

Overrides and  
Overloads

Comparison

Module Summary

## Inheritance in C++



# Inheritance in C++: Semantics

## Module M22

Partha Pratim Das

Objectives & Outlines

Inheritance in C++

Data Members

Object Layout

Member Functions

Overrides and Overloads

Comparison

Module Summary

- **Derived ISA Base**
- **Data Members**
  - **Derived** class *inherits* all data members of **Base** class
  - **Derived** class may *add* data members of its own
- **Member Functions**
  - **Derived** class *inherits* all member functions of **Base** class
  - **Derived** class may *override* a member function of **Base** class by *redefining* it with the *same signature*
  - **Derived** class may *overload* a member function of **Base** class by *redefining* it with the *same name*; but *different signature*
  - **Derived** class *may add* new member functions
- **Access Specification**
  - **Derived** class *cannot access private* members of **Base** class
  - **Derived** class *can access protected* members of **Base** class
- **Construction-Destruction**
  - A *constructor* of the **Derived** class *must first* call a *constructor* of the **Base** class to construct the **Base** class instance of the **Derived** class
  - The *destructor* of the **Derived** class *must* call the *destructor* of the **Base** class to destruct the **Base** class instance of the **Derived** class



# Data Members

## Module M22

Partha Pratim  
Das

Objectives &  
Outlines

Inheritance in  
C++

Data Members

Object Layout

Member  
Functions

Overrides and  
Overloads

Comparison

Module Summary

## Data Members



# Data Members

## Module M22

Partha Pratim Das

Objectives & Outlines

Inheritance in C++

Data Members

Object Layout

Member Functions

Overrides and Overloads

Comparison

Module Summary

- **Derived ISA Base**
- **Data Members**
  - **Derived** class *inherits* all data members of **Base** class
  - **Derived** class may *add* data members of its own
- **Object Layout**
  - **Derived** class *layout* contains an instance of the **Base** class
  - Further, **Derived** class *layout* will have data members of its own
  - C++ does not guarantee the *relative position* of the **Base** class instance and **Derived** class members





# Object Layout

## Module M22

Partha Pratim Das

Objectives & Outlines

Inheritance in C++

Data Members  
Object Layout

Member Functions

Overrides and Overloads

Comparison

Module Summary

```
class B { // Base Class
    int data1B_;
public:
    int data2B_;
    // ...
};
```

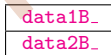
```
class D: public B { // Derived Class
    // Inherits B::data1B_
    // Inherits B::data2B_
    int infoD_; // Adds D::infoD_
public:
    // ...
};
```

```
B b; // Base Class Object
```

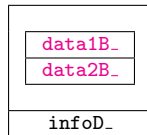
```
D d; // Derived Class Object
```

## Object Layout

Object b



Object d



- **d cannot access data1B\_ even though is a part of d!**
- **d can access data2B\_**



# Member Functions

## Module M22

Partha Pratim  
Das

Objectives &  
Outlines

Inheritance in  
C++

Data Members

Object Layout

**Member  
Functions**

Overrides and  
Overloads

Comparison

Module Summary

## Member Functions



# Member Functions

Module M22

Partha Pratim  
Das

Objectives &  
Outlines

Inheritance in  
C++

Data Members  
Object Layout

Member  
Functions

Overrides and  
Overloads

Comparison

Module Summary

- Derived **ISA** Base
- **Member Functions**
  - Derived class *inherits* all member functions of Base class
    - ▷ **Note:** Derived class *does not inherit* the Constructors and Destructor of Base class but *must have access to them*
  - Derived class may *override* a member function of Base class by *redefining* it with the *same signature*
  - Derived class may *overload* a member function of Base class by *redefining* it with the *same name*; but *different signature*
  - Derived class *may add* new member functions
- **Static Member Functions**
  - Derived class *does not inherit* the static member functions of Base class
- **Friend Functions**
  - Derived class *does not inherit* the friend functions of Base class



# Overrides and Overloads

## Module M22

Partha Pratim Das

Objectives & Outlines

Inheritance in C++

Data Members

Object Layout

Member Functions

Overrides and Overloads

Comparison

Module Summary

### Inheritance

```
class B { public: // Base Class
    void f(int i);
    void g(int i);
};
class D: public B { public: // Derived Class
    // Inherits B::f(int)

    // Inherits B::g(int)

};
B b;
D d;

b.f(1); // Calls B::f(int)
b.g(2); // Calls B::g(int)

d.f(3); // Calls B::f(int)
d.g(4); // Calls B::g(int)
```

- D::f(int) overrides B::f(int)
- D::f(string&) overloads B::f(int)

### Override & Overload

```
class B { public: // Base Class
    void f(int);
    void g(int i);
};
class D: public B { public: // Derived Class
    // Inherits B::f(int)
    void f(int); // Overrides B::f(int)
    void f(string&); // Overloads B::f(int)
    // Inherits B::g(int)
    void h(int i); // Adds D::h(int)
};
B b;
D d;

b.f(1); // Calls B::f(int)
b.g(2); // Calls B::g(int)

d.f(3); // Calls D::f(int)
d.g(4); // Calls B::g(int)

d.f("red"); // Calls D::f(string&)
d.h(5); // Calls D::h(int)
```



# Overloading vis-a-vis Overriding

## Module M22

Partha Pratim  
Das

Objectives &  
Outlines

Inheritance in  
C++

Data Members

Object Layout

Member  
Functions

Overrides and  
Overloads

Comparison

Module Summary

## Overloading vis-a-vis Overriding



# Comparison of Overloading vis-a-vis Overriding

## Module M22

Partha Pratim Das

Objectives & Outlines

Inheritance in C++

Data Members

Object Layout

Member Functions

Overrides and Overloads

Comparison

Module Summary

Basis	Function Overloading	Function Overriding
<b>Name of Function</b> <b>Signature</b> <b>Type of Function</b>	<ul style="list-style-type: none"><li>• All overloads have the same function name</li><li>• Function signatures must be different</li><li>• Can be global, friend, static or non-static member function</li></ul>	<ul style="list-style-type: none"><li>• All overrides have the same function name</li><li>• Function signatures are same</li><li>• Must be a non-static member function - non-virtual or virtual</li></ul>
<b>Inheritance</b> <b>Polymorphism</b> <b>Scope</b>	<ul style="list-style-type: none"><li>• Can happen with or without inheritance</li><li>• Static (Compile time)</li><li>• Overloaded functions are in the same scope</li></ul>	<ul style="list-style-type: none"><li>• Happens only with inheritance</li><li>• Static (Compile time) or Dynamic (Runtime)</li><li>• Functions are in different scopes (base class and derived class)</li></ul>
<b>Purpose</b>	<ul style="list-style-type: none"><li>• To have multiple functions with same name that act differently depending on parameters</li></ul>	<ul style="list-style-type: none"><li>• To perform additional or different tasks than the base class function</li></ul>
<b>Constructor</b> <b>Destructor</b> <b>Usage</b>	<ul style="list-style-type: none"><li>• Constructors can be overloaded</li><li>• The destructor cannot be overloaded</li><li>• Can be overloaded multiple times</li></ul>	<ul style="list-style-type: none"><li>• Constructors cannot be overridden</li><li>• The destructor cannot be overridden</li><li>• Can be overridden once in the derived class</li></ul>



# Module Summary

## Module M22

Partha Pratim  
Das

Objectives &  
Outlines

Inheritance in  
C++

Data Members

Object Layout

Member  
Functions

Overrides and  
Overloads

Comparison

Module Summary

- Discussed the effect of inheritance on Data Members and Object Layout
- Discussed the effect of inheritance on Member Functions with special reference to Overriding and Overloading