



## Module 23

Partha Pratim  
Das

Objectives &  
Outline

Inheritance in  
C++

protected  
Access

Constructor &  
Destructor

Object Lifetime

Summary

# Module 23: Programming in C++

## Inheritance: Part 3 (Constructor & Destructor - Object Lifetime)

Partha Pratim Das

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

*ppd@cse.iitkgp.ernet.in*

Tanwi Mallick  
Srijoni Majumdar  
Himadri B G S Bhuyan



# Module Objectives

## Module 23

Partha Pratim  
Das

Objectives &  
Outline

Inheritance in  
C++

protected  
Access

Constructor &  
Destructor

Object Lifetime

Summary

- Understand protected access specifier
- Understand the construction and destruction process on an object hierarchy
- Revisit Object Lifetime for a hierarchy



# Module Outline

## Module 23

Partha Pratim  
Das

Objectives &  
Outline

Inheritance in  
C++

protected  
Access

Constructor &  
Destructor

Object Lifetime

Summary

- ISA Relationship
- Inheritance in C++
  - Semantics
  - Data Members and Object Layout
  - Member Functions
    - Overriding
    - Overloading
  - protected Access
  - Constructor & Destructor
  - Object Lifetime
- Example – Phone Hierarchy
- Inheritance in C++ (private)
  - Implemented-As Semantics



# Inheritance in C++: Semantics

## Module 23

Partha Pratim Das

Objectives & Outline

Inheritance in C++

protected Access

Constructor & Destructor

Object Lifetime

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*
- **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



# Inheritance in C++:

## Access Members of Base: protected Access

Module 23

Partha Pratim  
Das

Objectives &  
Outline

Inheritance in  
C++

protected  
Access

Constructor &  
Destructor  
Object Lifetime

Summary

- Derived **ISA** Base
- Access Specification
  - Derived class *cannot access private* members of Base class
  - Derived class *can access public* members of Base class
- **protected** Access Specification
  - A new **protected** access specification is introduced for Base class
  - Derived class *can access protected* members of Base class
  - No other class or global function *can access protected* members of Base class
  - A **protected** member in Base class is like **public** in Derived class
  - A **protected** member in Base class is like **private** in other classes or global functions



# Inheritance in C++:

## protected Access

### Module 23

Partha Pratim  
Das

Objectives &  
Outline

Inheritance in  
C++

protected  
Access

Constructor &  
Destructor

Object Lifetime

Summary

#### private Access

```
class B {
private: // Inaccessible to child
        // Inaccessible to others
    int data_;
public:
    // ...
    void Print() { cout << "B Object: ";
                  cout<<data_<<endl;
    }
};

class D: public B { int info_;
public:
    // ...
    void Print() { cout << "D Object: ";
                  cout<<data_<<" "; // Inaccessible
                  cout<<info_<<endl;
    }
};

B b(0);
D d(1, 2);

b.data_ = 5; // Inaccessible to all

b.Print();
d.Print();
```

● **D::Print()** cannot access **B::data\_** as it is **private**

#### protected Access

```
class B {
protected: // Accessible to child
            // Inaccessible to others
    int data_;
public:
    // ...
    void Print() { cout << "B Object: ";
                  cout<<data_<<endl;
    }
};

class D: public B { int info_;
public:
    // ...
    void Print() { cout << "D Object: ";
                  cout<<data_<<" "; // Accessible
                  cout<<info_<<endl;
    }
};

B b(0);
D d(1, 2);

b.data_ = 5; // Inaccessible to others

b.Print();
d.Print();
```

● **D::Print()** can access **B::data\_** as it is **protected**



# Inheritance in C++: Streaming

## Module 23

Partha Pratim  
Das

Objectives &  
Outline

Inheritance in  
C++

protected  
Access

Constructor &  
Destructor

Object Lifetime

Summary

### Streaming in B

```
class B {
protected: int data_;
public:
    friend ostream& operator<<(ostream& os,
        const B& b) {
        os << b.data_ << endl;
        return os;
    }
};
class D: public B { int info_;
public:
    //friend ostream& operator<<(ostream& os,
    //    const D& d) {
    //    os << d.data_ << endl;
    //    os << d.info_ << endl;
    //    return os;
    //}
};

B b(0);
D d(1, 2);

cout << b; cout << d;

B Object: 0
B Object: 1
```

- d printed as a B object; info\_ missing

### Streaming in B & D

```
class B {
protected: int data_;
public:
    friend ostream& operator<<(ostream& os,
        const B& b) {
        os << b.data_ << endl;
        return os;
    }
};
class D: public B { int info_;
public:
    friend ostream& operator<<(ostream& os,
        const D& d) {
        os << d.data_ << endl;
        os << d.info_ << endl;
        return os;
    }
};

B b(0);
D d(1, 2);

cout << b; cout << d;

B Object: 0
D Object: 1 2
```

- d printed as a D object as expected



# Inheritance in C++: Constructor & Destructor

Module 23

Partha Pratim  
Das

Objectives &  
Outline

Inheritance in  
C++

protected  
Access

Constructor &  
Destructor

Object Lifetime

Summary

- **Derived ISA Base**
- Constructor-Destructor
  - Derived class *inherits* the Constructors and Destructor of Base class (*but in a different semantics*)
  - Derived class *cannot override* or *overload* a Constructor or the Destructor 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





# Inheritance in C++:

## Constructor & Destructor

### Module 23

Partha Pratim  
Das

Objectives &  
Outline

Inheritance in  
C++

protected  
Access

Constructor &  
Destructor

Object Lifetime

Summary

```
class B { protected: int data_;
public:
    B(int d = 0) : data_(d) { cout << "B::B(int): " << data_ << endl; }

    ~B() { cout << "B::~~B(): " << data_ << endl; }
    // ...
};

class D: public B { int info_;
public:
    D(int d, int i) : B(d), info_(i) // ctor-1: Explicit construction of Base
    { cout << "D::D(int, int): " << data_ << ", " << info_ << endl; }

    D(int i) : info_(i) // ctor-2: Default construction of Base
    { cout << "D::D(int): " << data_ << ", " << info_ << endl; }

    ~D() { cout << "D::~~D(): " << data_ << ", " << info_ << endl; }
    // ...
};

B b(5);
D d1(1, 2); // ctor-1: Explicit construction of Base
D d2(3); // ctor-2: Default construction of Base
```

### Object Layout

Object b

5

Object d1



Object d2





# Inheritance in C++:

## Object Lifetime

### Module 23

Partha Pratim  
Das

Objectives &  
Outline

Inheritance in  
C++

protected  
Access

Constructor &  
Destructor

Object Lifetime

Summary

```
class B { protected: int data_;
public:
    B(int d = 0) : data_(d) { cout << "B::B(int): " << data_ << endl; }
    ~B() { cout << "B::~~B(): " << data_ << endl; }
    // ...
};

class D: public B { int info_;
public:
    D(int d, int i) : B(d), info_(i) // Explicit construction of Base
    { cout << "D::D(int, int): " << data_ << ", " << info_ << endl; }
    D(int i) : info_(i) // Default construction of Base
    { cout << "D::D(int): " << data_ << ", " << info_ << endl; }
    ~D() { cout << "D::~~D(): " << data_ << ", " << info_ << endl; }
    // ...
};

B b(0);
D d1(1, 2);
D d2(3);
```

#### Construction O/P

```
B::B(int): 0           // Obj. b
B::B(int): 1           // Obj. d1
D::D(int, int): 1, 2   // Obj. d1
B::B(int): 0           // Obj. d2
D::D(int): 0, 3        // Obj. d2
```

#### Destruction O/P

```
D::~~D(): 0, 3        // Obj. d2
B::~~B(): 0           // Obj. d2
D::~~D(): 1, 2        // Obj. d1
B::~~B(): 1           // Obj. d1
B::~~B(): 0           // Obj. b
```

- First construct base class object, then derived class object
- First destruct derived class object, then base class object



# Module Summary

## Module 23

Partha Pratim  
Das

Objectives &  
Outline

Inheritance in  
C++

protected  
Access

Constructor &  
Destructor

Object Lifetime

Summary

- Understood the need and use of protected Access specifier
- Discussed the Construction and Destruction process of class hierarchy and related Object Lifetime



# Instructor and TAs

## Module 23

Partha Pratim  
Das

Objectives &  
Outline

Inheritance in  
C++

protected  
Access

Constructor &  
Destructor

Object Lifetime

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655