

FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

Lecture11

Lecture 11: Access Specifiers in C++

Access Specifiers in C++

```
class MyClass {  
public:  
    int publicVar;        // can be accessed from anywhere  
    void publicFunc() { // can be accessed from anywhere  
        cout << "This is a public function" << endl;  
    }  
private:  
    int privateVar;      // can only be accessed from within the class  
    void privateFunc() { // can only be accessed from within the class  
        cout << "This is a private function" << endl;  
    }  
}
```



So, welcome to Lecture 11. So, in this lecture, we are going to see the new chapter called inheritance. So, we had seen in the introductory class, we had seen the basics of inheritance. Because we are going to use the code reusability. That means, assume that somebody has created Class A, and you are the new programmer.

And you created Class B, and you want to use the properties of Class A. Right. So, in that case, is it possible? Yes, it is possible. Right.

And the concept is called inheritance. So, before going into the details of inheritance. All right. So, still we have not seen the keywords public, private, and protected. Right.

So this particular example will tell you. So what is meant by the public access specifier or access modifier? What do you mean by private? What do you mean by protected? So let us consider my class, right?

Let us consider the class, my class. So here you have public member data and public member functions. So what do you mean by public member data? So this can be accessed from anywhere. This particular data or member data can be accessed from anywhere in the program.

Similarly, you have a void public function. So this member function can be accessed from anywhere in the program. So it has one 'cout' statement. This is a public function. And another access specifier is private.

You have a private variable. So this can be accessed only within the class. So inside the class, that means MyClass can use this. So here you have MyClass. So inside this MyClass, you can use this.

So when you are creating an object outside the class, assume that you are creating an object, right? MyClass, let us say C1. The C1 cannot access this private variable, and similarly, it cannot access a private function, right? You are defining it outside the class. So that means when you have the private member data,

Or a private member function. So this can be accessed. Only within the class. Right. So you cannot access it outside the class.

So here. The function. The function is a private function. cout. This is a private function.

Right. And similarly. You have protected. Access specifier. Right.

Or access modifier. Protected. Right? So private and public we have seen so far. First time we are seeing protected.

So, what do you mean by protected? Suppose I have a variable name like this. Right? It is almost similar to private. But the only difference is the derived class can access this.

Right? The derived class. The base class and derived class. If you remember, we had seen the introductory inheritance. Right.

Assume that I have class A. So when I put class A, publicly inheriting class B, something like this. Right. So, either I can put public or private. Now we are studying protected. So when you have the protected member data.

Right. So assume that when I write like this, like a private public inside the class. So, in fact, this is inside the class. Correct. So this is inside the class.

```
protected:
    int protectedVar;
    //can only be accessed from within the class and its
    // derived classes
    void protectedFunc(){// can only be
    //accessed from within the
    //class and its derived classes
        cout << "This is a protected function" << endl;
    }
};
```

Handwritten notes and diagram:

- Class B : public Class A
- Class C : public Class B
- Arrows indicate inheritance from Class A to Class B, and from Class B to Class C.
- A small video inset shows a man in a blue shirt looking at a screen.

As I said, private member data, I cannot access at all. But whereas protected member data is almost like private, the only difference is the derived class can access it. So now when I put class B, class B can access the protected variable. Right. The protected var.

Whereas class B cannot access your private variable. Right. So when I am inheriting. Correct. So when I am inheriting.

So assume that class A is nothing but MyClass. Right. Here, this A is nothing but MyClass. From a syntax point of view, I wrote it like this. Correct.

So when it is protected, it can be accessed in the derived class. Whereas private member data cannot be accessed by the derived class. That is the only difference. The derived classes can access the protected member data or even the member functions. So next, we are seeing the member functions.

So this can be accessed. The member functions can be accessed. Right. By the derived class. And then the protected function you are writing.

This is a protected function. And the class is over. So now we have seen all the access modifiers. Now you know the meaning, right? So we have completed objects, classes, and constructors.

We have kept on using public and private. Slowly we are understanding the access specifiers: public, private, and protected. So public means it can be accessed anywhere. So this member data and member functions can be accessed anywhere. Whereas if it is private, it can be accessed only within the class.

And in the case of protected, it can be accessed by the derived class only. It is almost like private, right? So, we are going to study. Once the derived class is accessing, so for the derived class, the protected data member will become private, right? In the derived class B, this protected member data will be private.

So, suppose I am going for the multi-level class C. Right? Class C is publicly inheriting. So, let us say class. In fact, I have to write like this. Class B. This is just an example I gave previously.

Here you go. Class B is publicly inheriting class A. Right? And here, class C is publicly inheriting class B. So, in this case, what is happening? Class A, assume that class A is protected, right?

Class A contains the protected member data. So, in class B, it is becoming private. It can access, but it is becoming private. So, C cannot access now, right? So, that means B is a derived class of A, and it can access.

Right? So that is what is written over here. It can only be accessed from within the class and its derived classes. And its derived classes. So B is a derived class of A, but C cannot access.

Right? So that is the meaning. So it is almost like a private. So you can see how the data is being secured. Right?

So whatever member data you are using, see how it is being secured. So now we will see an example. All right. So we are using several public member data and member functions. All right.

The public Members

- A public member is accessible from anywhere outside the class but within a program.

```
1  #include <iostream>
2  using namespace std;
3
4  class Line {
5      public:
6          double length;
7          void setLength( double len );
8          double getLength( void );
9  };
10
11  // Member functions definitions
12  double Line::getLength(void) {
13      return length ;
14  }
15
```



So, first, we will talk about the public member data and member functions. So, that means this can be accessed anywhere outside the class. All right. But within a program. That is obvious.

Correct. So, you have to use it within a program. So, now I have a class line. All right. Assume that I have a class line.


So, the class line contains a member data called length. Fine. And then, you have two member functions. Alright. So, one is setLength and getLength.

The setter function. So, in fact, you are studying about the setter function and the getter function. setLength and getLength, you can simply put void. Void means nothing. Alright.

It is equivalent to getLength without putting anything. So, that is also valid. Alright. And here I am. Defining right outside the class was possible. Yesterday, we had seen it right in the last class. We had seen it correct.

So, line number 12, you have double Line, right? So, Line is the name of the class. You have scope resolution operator, getLength is the member function you are defining here. It is a simple function returning length. Right. It is a simple function that is returning length. And then, another function called setLength.


```
19 void Box::setWidth( double wid ) {  
20     width = wid;  
21 }  
22  
23 // Main function for the program  
24 int main() {  
25     Box box;  
26     // set box length without member function ✓  
27     box.length = 10.0; // OK: because length is public  
28     cout << "Length of box : " << box.length << endl;  
29  
30     // set box width without member function  
31     // box.width = 10.0; // Error: because width is private  
32     box.setWidth(10.0); // Use member function to set it.  
33     cout << "Width of box : " << box.getWidth() << endl;  
34     return 0;  
35 }
```



So, line dot getLength. What is getLength? This is a member function. So, which will be invoked. So, here you go.

getLength. Right. So, length we already got it is 6.0 from here, right. So, 6.0, so return length, so 6.0 will be printed, right. So, it is returning length.

So, length is nothing but you are passing 6.0, so that will be printed. So, the first output cout, I should get 6.0, we will see, right. So, second I am just Without member function, what am I doing? So, this I can do that, right?

Without member function, I am changing the length. The line dot length, that means I can access the member data. That is what I want to say, right? I can access member data because that is public line. Number 5, if you look, it is public.

So, double length, I am able to access this. And then I am writing line dot length, which is possible in the main, equal to 10.0. So it will work because length is public. So now I am trying to print the length of the line. Line dot length.

Right. So line is an object, and the dot operator is invoking the member data length. That is the meaning. Right. So 10.0 will be printed for line number 13.

Right. The program ends. Return 0. Program ends. So, what is the output you are getting?

6 and 10. So, we are getting the output 6 and 10. I hope it is clear for everyone. So, you can access the public member data and member functions anywhere, right? Within the class or outside the class, but of course, within the program, right? So, public member data and member functions can be accessed everywhere.

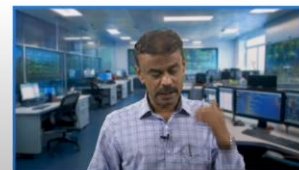
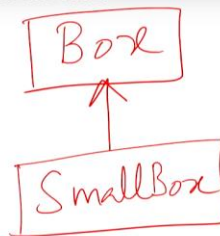
Okay. So, with this knowledge, we will see private member data. We can see an example of private member data. So, when I have private member data and private member functions, right? So, this cannot be accessed or even viewed outside the class, right?

Only the class, and sooner or later, we are going to talk about the friend function, right? Only the This particular class where you are creating and then writing this private member data and member function can access, or we are going to study friend function that can access the private member data and member functions. So, now we can see this example. Alright.

So, you have public length that can be accessed anywhere outside the class, and setWidth can be accessed everywhere. getWidth can be accessed everywhere. Whereas you have one private member data. We have to be a little careful. Right.

The protected Members

```
1  #include <iostream>
2  using namespace std;
3
4  class Box {
5      protected:
6          double width;
7  };
8
9  class SmallBox:Box { // SmallBox is the derived class.
10     public:
11         void setSmallWidth( double wid );
12         double getSmallWidth( void );
13     };
14
```



So this can be accessed only within the class. Right. So here you have the class is over. But whereas you are defining two functions. Get width, line number 8, you are defining.

So this is as such when you are putting a scope resolution operator; it is as such you are writing this particular function inside the class. Right? Inside the class box. Right? The meaning is inside the class box, I am writing this.

That is the meaning. Right? So it is returning width. Not a problem. This can access.

Right? Because it is inside the class. Return width. Width can be accessed by the getWidth member function. And similarly, here you have

setWidth, which is also inside line number 7, right? And whatever you are passing will be assigned to width, that is the meaning, right? So with this knowledge, we can go inside, right? The main program. So you are creating a box, small b, which is the object, and capital B is the class, right?

```
15 // Member functions of child class
16 double SmallBox::getSmallWidth(void) {
17     return width ;
18 }
19 void SmallBox::setSmallWidth( double wid ) {
20     width = wid;
21 }
22
23 // Main function for the program
24 int main() {
25     SmallBox box;
26
27     // set box width using member function
28     box.setSmallWidth(5.0);
29     cout << "Width of box : " << box.getSmallWidth() << endl;
30
31     return 0;
32 }
```



So now, let us put. Box dot length is equal to 10.0. All right. Box dot length is equal to 10.0. Absolutely fine.

Because length is public. All right. The length member data is public. Not a problem. Right.

And then cout length of the box. So obviously I will get 10 as output. First output will be 10. So next I am trying to. Right.

Suppose this comment statement is not there. Assume that. Box dot width is equal to 10.0. Right. What will happen?

I will get an error. Why am I getting an error? Because width is private to the class Box. So the private member data I cannot access. Right.

So what can we do with the same code? We remove this comment statement and try to run the code. You will get an error. Right? So this is the meaning.

We cannot access the private member data and private member functions. So this will throw an error. So now, how can we use it? So, for example, I call, I mean the box is invoking setWidth. The box is calling setWidth by passing 10.0.

So when I am passing 10.0, right, your WID is assigned to width. Now it can be accessed. Right. But you have to use it in a careful way. You are using a setter function, and from the setter function, you are trying to access width, which is possible.

Correct. So you are passing 10.0, and there, width is stored as 10.0. Whereas directly, you cannot do that. This cannot be done. This is perfectly right.

Right. So now you are trying. So 10.0 I put width will be 10.0. Width is equal to WID 10.0. So the box object whose width is 10.0.

So now you are trying to print this. Cout width of the box. Right. So box is invoking getWidth. Right.

Box is invoking the getWidth. So where is your getWidth? Here you go. Return width. All right.

So, width is assigned to 10.0 here. Correct. Correct. So, that will be printed. So, I will get one more 10 as the output.

Correct? So, when I run the code, I will get the output. So, the length of the box is 10, and the width of the box = 10. Right? So, this is what the output we are expecting, and we are getting.

Right? So, the problem here is, if you look at line number 31, assume that there is no common statement, and then if you try to run the code, you will get an error. The reason is, Outside the class, you cannot access the private member data. Right.

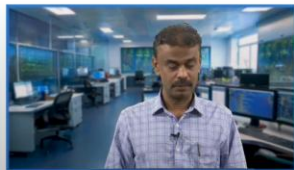
So these are the outputs we have got. So this is how we can use the private member data and the public member data. In the sense, member function as well. Okay. So now, protected.

The protected Members

```
1  #include <iostream>
2  using namespace std;
3
4  class Box {
5      protected:
6          double width;
7  };
8
9  class SmallBox:Box { // SmallBox is the derived class.
10     public:
11         void setSmallWidth( double wid );
12         double getSmallWidth( void );
13     };
14
```

Diagram illustrating class inheritance:

- Box** → Base class
- SmallBox** → Derived class



So protected, as I already said, is very similar to private member data. Right. Private member. But the only difference is the derived class can access, right? You have base class and derived class.

The derived class can access the private member data. So for this, all right, so maybe I will briefly tell about this. Slowly, I am introducing the inheritance. So what do you mean by inheritance? So you will see that, right?

So, in fact, we had seen an example, right? If you recall in the introductory class, we had seen the example, right? How to use inheritance. So, assume that I have a class Box, right? Box is the base class.

Box is the base class, right? It has one protected member data called width, right? One protected member data called width. And I have another box called SmallBox, right? So, now the syntax here is right: SmallBox colon Box.

So, that means it is inheriting. That means it can use the property. That means the member data and member functions of Box. That is the meaning. We studied this during our introductory class.

If you recall, I introduced the inheritance. We have five different inheritances. I introduced this. Right. So now, in brief, we are going to see in this chapter, we are going to see in brief.

So SmallBox, that means SmallBox is the derived class, and which is the base class? Box is the base class, right. So now you have public member functions in SmallBox: setSmallWidth, getSmallWidth, right. You have one argument width in setSmallWidth, and it is a void for, I mean, no argument in getSmallWidth.

Your Box class is ending in line number 7, and SmallBox class is ending in line number 13. So now we can see how we can access this width. How SmallBox is going to access this protected member data called width. So here we define the SmallBox class. getSmallWidth, getSmallWidth is line number 12, and setSmallWidth, which is line number 11, right, which is line number 11.

So, now you go to the main program, right? So, here, like the last program, the get and set, right? Getter function and setter function, you call it, is a very famous one in object-oriented, right? So, you are setting, like in the last two programs, you are doing this. So, now I create an object box under the SmallBox class, is SmallBox, all right. So, now I try to access setSmallWidth.

Alright. I try to access setSmallWidth. Alright. SetSmallWidth over here, line number 11. SetSmallWidth.

That means you have width WID. So, that will be assigned to width. Alright. That will be assigned to width. So, now inside the class.

Right. Obviously, inside the class, it can be accessed. Width can be accessed. Not a problem. Right, so 5.0 will be set over there, correct? So that will be set over there. So, it is width and getSmallWidth will return width, right? It will return width, right? So, what is the value?

5.0, 5.0 will be printed, right? Or even you can try, right? Now you can change the value. Right. So, what you can do is, you can put this as width only, right? You can try, right? You can try to put, right, outside the class. You can try to put,

let's say, `box.width`, you will get an error, right. So, it cannot be accessed outside the class. It is behaving exactly like a private member data, whereas Since `SmallBox` is the derived class.

Alright. Since `SmallBox` is the derived class. It can access. Alright. It can access `width`.

So, we are trying to access `width` in two cases. Alright. So, when I am setting. `SetSmallWidth 5.0`. Alright.

So, `WID`. I am passing `5.0`. Your `WID` is `5.0`. So now, `width` is equal to `5.0`. Possible, right? So that means `SmallBox` is a derived class that can access the protected member data of the base class, right? Another one, you are returning, right?

When I am calling `getSmallWidth`, it is returning `width`. Yes, this particular member function can also access because this member function is under the derived class, right? Whereas, if you try to put So, assume that you are trying to put, let us say, `cout << box dot width`, it will give an error. Right.

So similarly, you will get an error if it is private. Assume that line number 5. What do you do? So you can test now. You can test now various cases. So slowly, I mean, you have already studied classes, objects, constructors, etc.

So now you can practice. So remove line number five. What do you do? Remove protected to private. Right.

And then you can check. So, in fact, the derived class, what we have studied previously, the derived class cannot access. Right. Because it is private. Correct?

The derived class cannot access it. So, it will throw an error. Line number 17 will have an error. Line number 20 will have an error. If you use private instead of protected, if you use private.

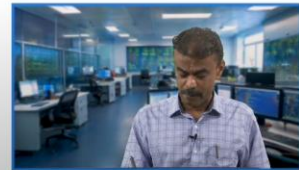
Right? So, that is the difference between protected and private. Whereas, since I put protected in line number 5, this `SmallBox` class has the privilege to use `width` in line number 17 and line number 20. Okay, but whereas outside this class, outside this derived class, suppose in the main program, I try to write, let us say, `box dot width equal to`, let us say, `5.0`. So then what will happen? I will get an

error. So now when I run the code, I have to get an output: width of the box is 5, right? So this is what we expected, so we are getting this. This is how the protected access specifier is working.

Inheritance

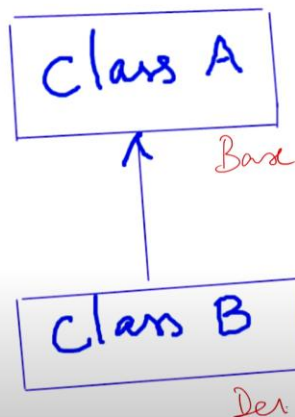
- Inheritance is a key feature of Object-Oriented Programming (OOP).
- It allows a class (called the derived class) to inherit attributes and functions from another class (called the base class).
- This promotes code reusability and helps to establish relationships between classes.
- Through inheritance, we achieve function overriding as well.

Class A {
...
}
Class B: public



So now, at the initial point of time, I talked about the basics of inheritance. Right? So it is a very key feature of object-oriented programming. Suppose someone has written the code A. Right? Class A. So that means he or she might have used various member data and member functions.

Single/Simple Inheritance



Single inheritance is defined as the inheritance in which a derived class is inherited from the only one base class.

Here 'A' is the base class, and 'B' is the derived class.



Right? So, various member data and member functions. So, now I am trying to modify the code. Right? I am writing class B. Right?

So, can I use the properties of class B? So, can I publicly inherit class A? Yes, it is possible. That is called inheritance. Right?

So, one is called the base class. So, for example, here A is the base class. Right? And B is the derived class. So that means B is using the properties of, right, properties meaning all the member data and member functions.

So that we can call it as it inherits all the member data and member functions, or it is inheriting all the attributes and functions, right. So one you call it as base, another one is derived. In Java, it is called a superclass. A is a superclass, and B is a subclass. So we will see that.

We will see those notations as well. In Java, you call A a superclass and B a subclass or child class. So this is what I said about reusability. I can use the properties of class A when I am writing the new class B. So it can take all the attributes and functions. So that is the advantage of inheritance.

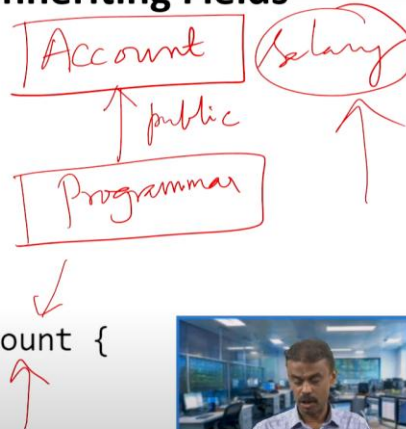
And also we are going to study the concept of overriding. Overriding means, assume that class A, you may have, let us say, a function, right? The name of the function is the same, int fun1, all right? And class B also has the same function, all right?

So, in that case, when you are creating an object and that object is invoking this fun1, which one will be called, right? So, this will be decided during runtime. In fact, I explained about this, right. So, the overriding concept we will see with the help of inheritance. So, here this diagram we are already familiar with, right.

So, class A is called the base class and this is the derived class, right. When I want to define a single inheritance or simple inheritance, right. So, here you have the derived class B, right. Which is inherited from the base class. Right.

Single Level Inheritance: Inheriting Fields

```
1  #include <iostream>
2  using namespace std;
3
4  class Account {
5      public:
6      float salary = 60000;
7  };
8  class Programmer: public Account {
9      public:
10     float bonus = 5000;
11 };
12
```



Which is inheriting from the base class. So you have only one base class. And then class B is inheriting all the properties of all the properties, in the sense attributes and methods. Or member functions. Member data and member functions.

So, B can use. So again, we are going to use the public, private, and protected. Right, you are inheriting publicly, you are inheriting privately, what will happen? So, these things we are going to study, and here A is the base class and B is the derived class. So, this inheritance is called the simple inheritance or single-level inheritance because we will also study the multilevel.

So, this is called the single or simple inheritance. We will take this example, right. So, assume that I have a class Account. All right. I have a class Account.

So, Account is the base class. You can draw a diagram. Account is the base class. And you have one member data, which is a public member data. So now, you'll slowly understand what is public, private, and protected.

Member data salary is 60,000. Your class is over here. Your class Account is over here. So that means it contains salary. Remember the data called salary.

So now you have another class called Programmer. And you have this is the syntax. You put a colon. I am going to publicly inherit the Account class. All right.

You can write it here as public. So this is the notation you are using. So that means Programmer is publicly inheriting the Account class. The Programmer

class is publicly inheriting the Account class. The Account class contains only one member data called salary.

So, that means whenever you are creating an object under Programmer, that object can access salary. Right. That is the meaning. So, your class Programmer. So, publicly write in editing Account class.

And this Programmer has one member data called bonus. Right. So, it is having member data called bonus. Class ends here. That means the Programmer class ends here.

Line number 11. And here you go. So, I have the object p1. Under the class Programmer. So we are.

Alright. We are printing out. Two things here. p1 dot salary. So p1 is accessing salary.

Alright. So the salary is 60,000. p1 was created under. Programmer. Right.

So p1 was created under class Programmer. So this p1. Can access. So p1 is an object. Because of the public inheritance, p1 can access salary.

So, what are you getting? p1.salary. The first output I expect is 60,000. And the rest is the same. The same class you are accessing.

p1.bonus. Bonus which is available over here. Line number 10. 5,000. So, I will get 5,000 as output.

Correct. So, when I run the code, I will get a salary of 60,000. This is what we are expecting for the first output and a bonus of 5,000. So, the point here is the object p1 you are creating under Programmer. Right.

In this particular class, that means a derived class. Right. This is a derived class. Right. So, in the derived class, you have p1.

So, this p1 can access the base class properties. Right, properties mean the member data and member functions, so it can access salary as well. Therefore, when you are printing p1.salary, you are getting the output. So, this inheritance is called single or simple inheritance. Right? In the next few classes, we are going to see many other four other inheritance types in C++ and also we will study The

same inheritance concept in Java. So, Java is slightly tricky. So, there are some inheritances that we cannot do with the help of classes.

So, that also we will see in the next class. Thank you.