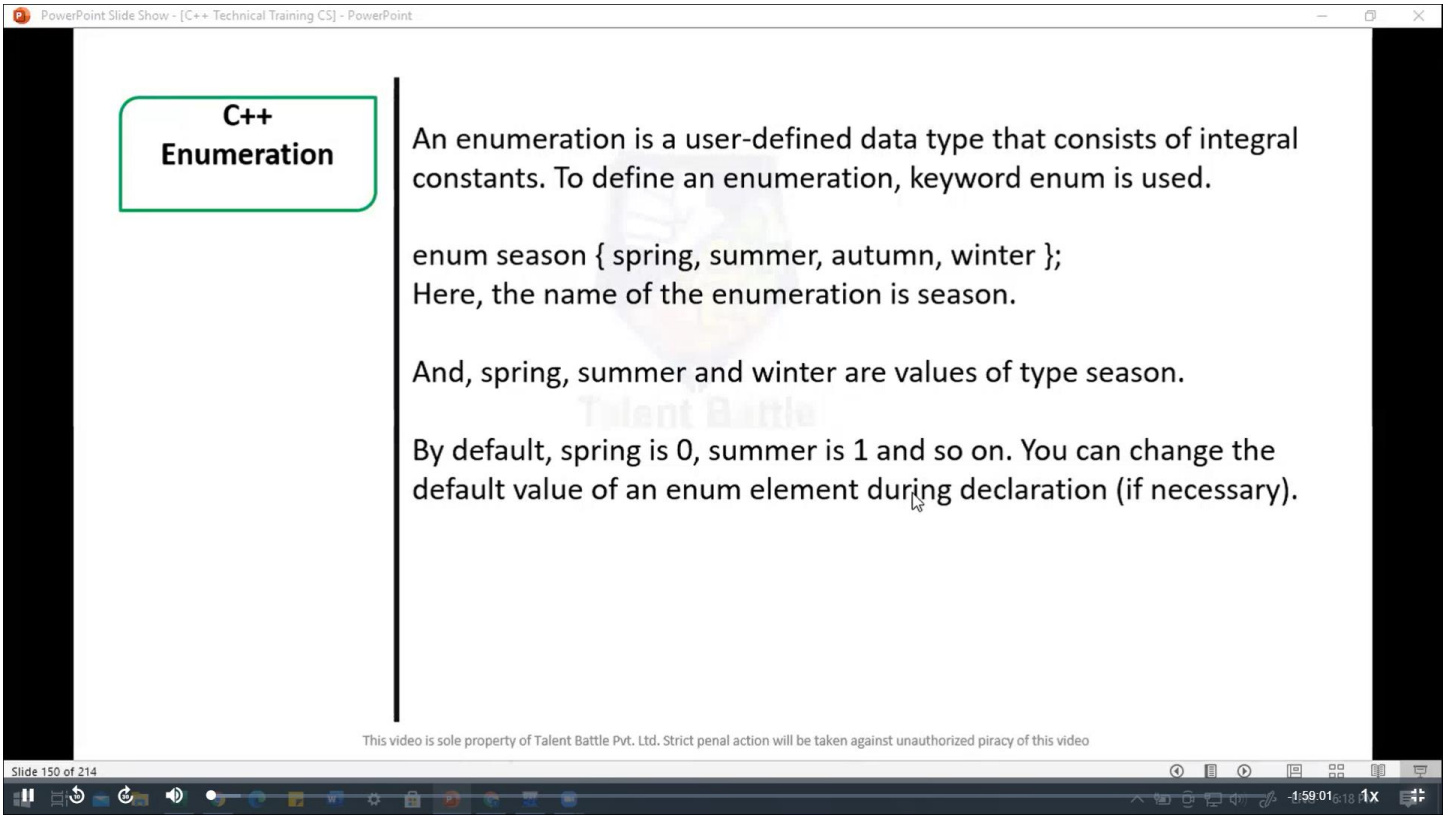
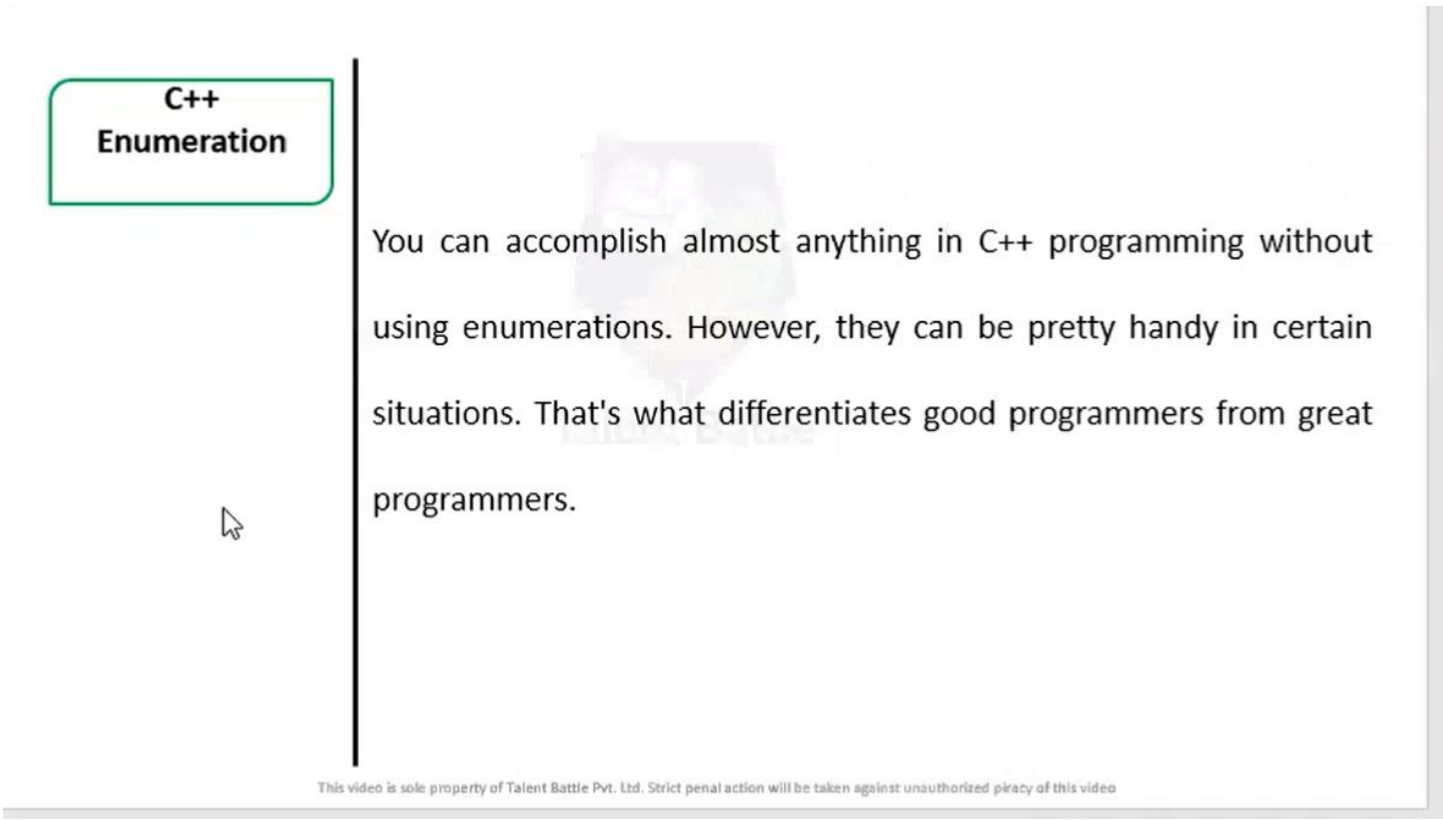
Day4

C++ Class & Object




```cpp
// Enumeration
#include <iostream>
using namespace std;

enum week { Sunday, Monday, Tuesday, Wednesday, Thrsday, Friday, saturday
};

int main(){
  week today;
  today = Wednesday;
  cout << "Day" << today + 1;
  return 0;
}

/*
Day4
--------------------------------
Process exited after 0.1415 seconds with return value 0
*/
```

```cpp
//============================================================
// Enumeration ex: 2

#include <iostream>
using namespace std;

enum seasons{
  spring = 34,
  summer = 4,
  automn = 9,
  winter = 32
};

int main(){
  seasons s;
  s = summer;
  cout << "Summer = " << s << endl;
  return 0;
}

/*
Summer = 4

--------------------------------
Process exited after 0.01113 seconds with return value 0
*/
```



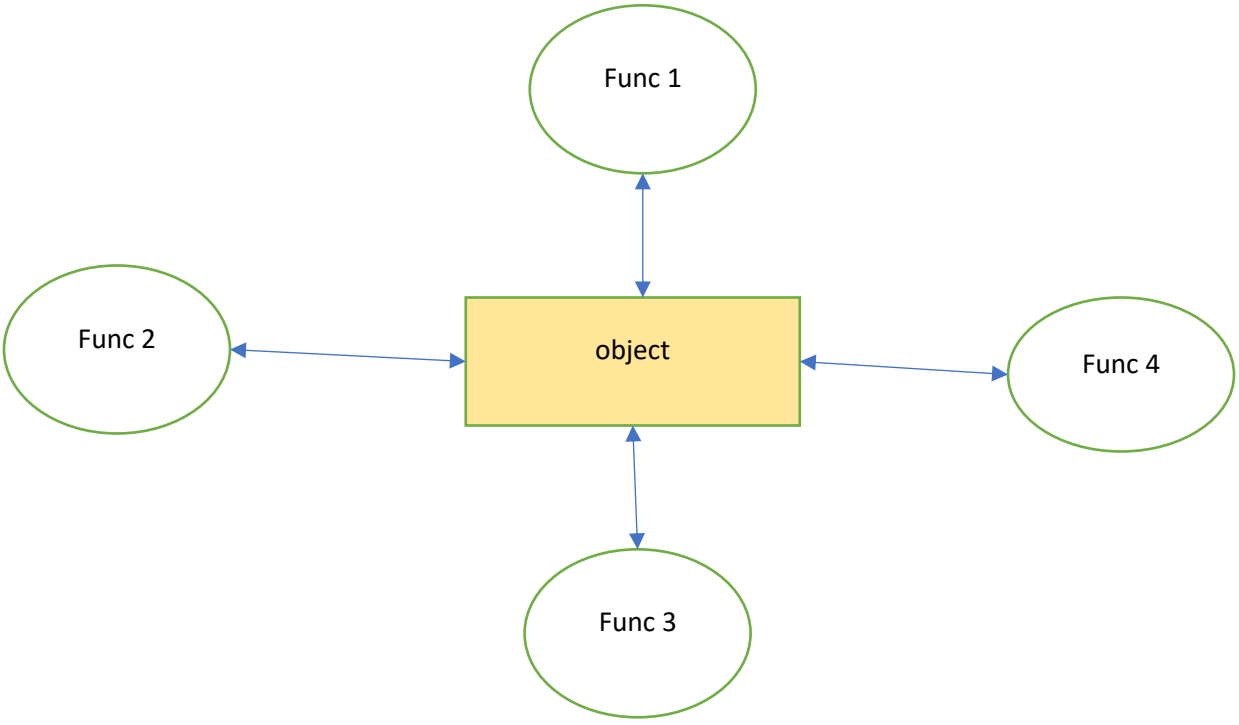
C++ Objects & Class

Suppose, we need to store the length, breadth, and height of a rectangular room and calculate its area and volume.

To handle this task, we can create three variables, say, length, breadth, and height along with the functions calculateArea() and calculateVolume().

However, in C++, rather than creating separate variables and functions, we can also wrap these related data and functions in a single place (by creating objects). This programming paradigm is known as object-oriented programming.

But before we can create objects and use them in C++, we first need to learn about classes.



This is Object oriented programming

PowerPoint Slide Show - [C++ Technical Training CS] - PowerPoint

C++ Objects & Class

C++ Class

A class is a blueprint for the object.

We can think of a class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows, etc. Based on these descriptions we build the house. House is the object.

Create a Class

A class is defined in C++ using keyword class followed by the name of the class.

The body of the class is defined inside the curly brackets and terminated by a semicolon at the end.

```
class className {
  // some data
  // some functions
};
```

For example,

```
class Room {
   public:
      double length;
      double breadth;
      double height;
      double calculateArea(){
         return length * breadth;
      }
```


Slide 155 of 214

-1:23:28 1.5x

PowerPoint Slide Show - [C++ Technical Training CS] - PowerPoint

C++ Objects & Class

C++ Class

A class is a blueprint for the object.

We can think of a class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows, etc. Based on these descriptions we build the house. House is the object.

Create a Class

A class is defined in C++ using keyword class followed by the name of the class.

The body of the class is defined inside the curly brackets and terminated by a semicolon at the end.

```
class className {
  // some data
  // some functions
};
For example,
class Room {
   public:
      double length;
      double breadth;
      double height;
      double calculateArea(){
         return length * breadth;
      }
```

Access Specifier

Mem. var

Mem. func.

This video is sole property of Talent Battle Pvt. Ltd. Strict penal action will be taken against unauthorized piracy of this video

Slide 155 of 214

-1:17:11 1x

C++ Objects & Class

Syntax to Define Object in C++

className objectVariableName;

We can create objects of Room class (defined in the above example) as follows:

```
// sample function
void sampleFunction() {
   // create objects
   Room room1, room2;
}
int main(){
   // create objects
   Room room3, room4;
}
```

Here, two objects room1 and room2 of the Room class are created in sampleFunction(). Similarly, the objects room3 and room4 are created in main(). As we can see, we can create objects of a class in any function of the program. We can also create objects of a class within the class itself, or in other classes. Also, we can create as many objects as we want from a single class.

C++ Objects & Class

C++ Access Data Members and Member Functions

We can access the data members and member functions of a class by using a . (dot) operator. For example,

```
room2.calculateArea();
```

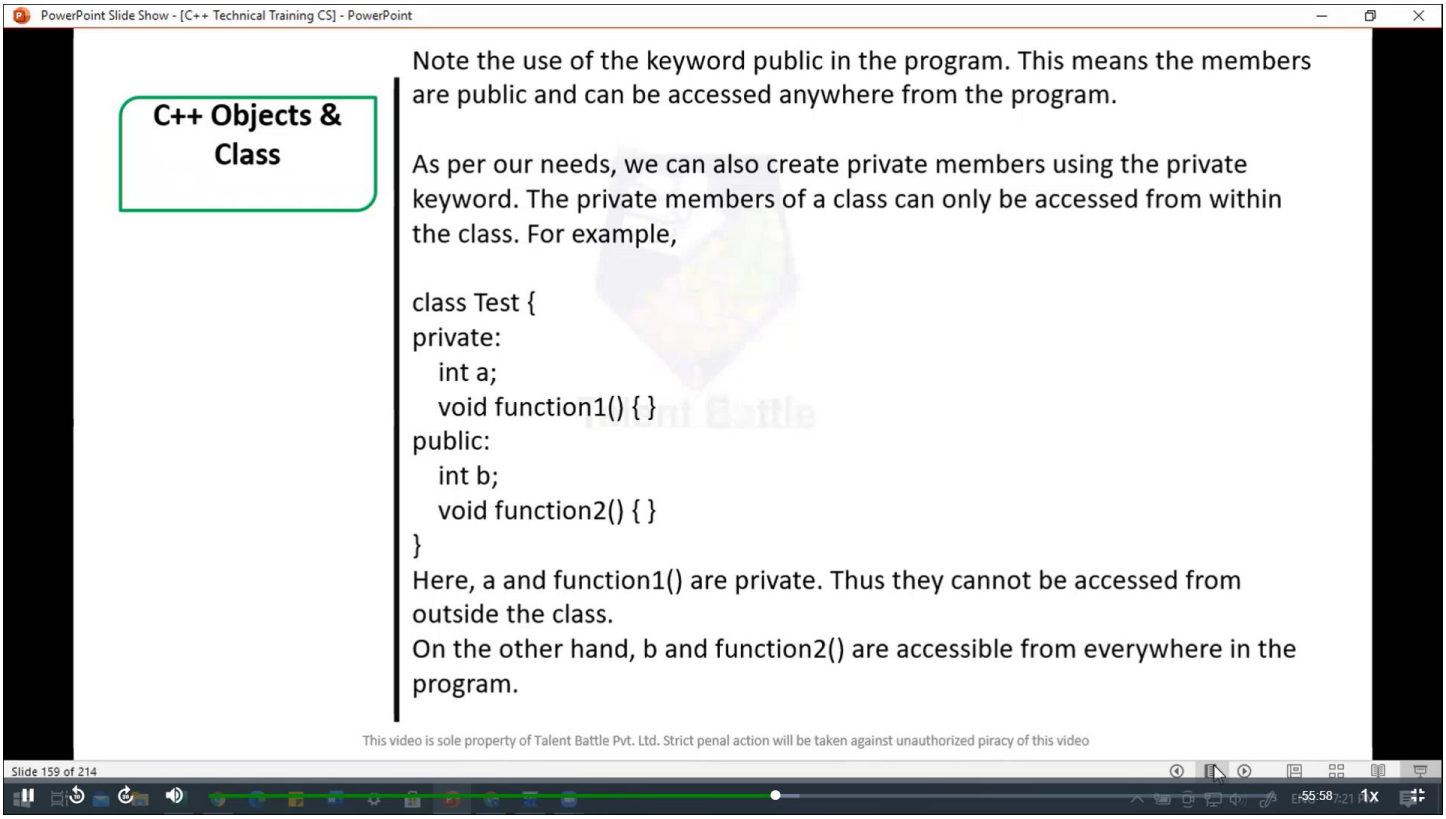
This will call the calculateArea() function inside the Room class for object room2.

Similarly, the data members can be accessed as:

```
room1.length = 5.5;
```

In this case, it initializes the length variable of room1 to 5.5.

```cpp
//============================================================
// C++ Objects & Class
// program to illustrate the working of objects and
// class in c++ programming

#include <iostream>
using namespace std;

// create a class
class Room {
  public:
    double length;
    double breadth;
    double height;

    double calculateArea(){
      return length * breadth;
    }

    double calculateVolume(){
      return length * breadth * height;
    }
};

int main(){
  // create object of Room  class
  Room room1;

  // assign values to data members
  room1.length = 42.5;
  room1.breadth = 30.8;
  room1.height = 19.2;

  // calculate and display the area and volume of the room
  cout << "Area of Room = " << room1.calculateArea() << endl;
  cout << "Volume of Room = " << room1.calculateVolume() << endl;

  return 0;
}


/*
Area of Room = 1309
Volume of Room = 25132.8

--------------------------------
Process exited after 0.1369 seconds with return value 0
Press any key to continue . . .
*/
```


PowerPoint Slide Show - [C++ Technical Training CS] - PowerPoint
C++ Objects & Class
Note the use of the keyword public in the program. This means the members are public and can be accessed anywhere from the program.
As per our needs, we can also create private members using the private keyword. The private members of a class can only be accessed from within the class. For example,
class Test {
private:
int a;
void function1() { }
public:
int b;
void function2() { }
}
Here, a and function1() are private. Thus they cannot be accessed from outside the class.
On the other hand, b and function2() are accessible from everywhere in the program.
This video is sole property of Talent Battle Pvt. Ltd. Strict penal action will be taken against unauthorized piracy of this video
Slide 159 of 214
55:58
1x

```cpp
//=============================================
// Program to illustrate the working of Public and private in C++ class

#include <iostream>
using namespace std;

class Room{
  private:
    double length;
    double breadth;
    double height;

  public:
    // function to initialize private variables
    void getData(double len, double brth, double hgt){
      length = len;
      breadth = brth;
      height = hgt;
    }

    double calculateArea(){
      return length * breadth;
    }

    double calculateVolume(){
      return length * breadth * height;
    }
};

int main(){
  // create object of Room class
  Room room1;
  // pass the values of private variables as arguments
  room1.getData(42.5, 30.8, 19.2);

  cout << "Area of Room = " << room1.calculateArea() << endl;
  cout << "Volume of Room = " << room1.calculateVolume() << endl;

  return 0;
}


/*
Area of Room = 1309
Volume of Room = 25132.8

--------------------------------
Process exited after 0.06076 seconds with return value 0
*/
```

C++ Objects & Class

Since the variables are now private, we cannot access them directly from main(). Hence, using the following code would be invalid:

```
// invalid code
obj.length = 42.5;
obj.breadth = 30.8;
obj.height = 19.2;
```

Instead, we use the public function getData() to initialize the private variables via the function parameters double len, double brth, and double hgt.

C++ Objects & Class

C++ Access Modifiers

One of the main features of object-oriented programming languages such as C++ is data hiding.

Data hiding refers to restricting access to data members of a class. This is to prevent other functions and classes from tampering with the class data.

However, it is also important to make some member functions and member data accessible so that the hidden data can be manipulated indirectly.

The access modifiers of C++ allows us to determine which class members are accessible to other classes and functions, and which are not.

C++ Objects & Class

For example,

```
class Patient {
   private:
      int patientNumber;
      string diagnosis;
   public:
    void billing() {
       // code
    }
    void makeAppointment() {
       // code
    }
};
```

Here, the variables patientNumber and diagnosis of the Patient class are hidden using the private keyword, while the member functions are made accessible using the public keyword.

C++ Objects & Class

Types of C++ Access Modifiers

In C++, there are 3 access modifiers:

- public
- private
- protected

PowerPoint Slide Show - [C++ Technical Training CS] - PowerPoint

C++ Objects & Class

public Access Modifier

The public keyword is used to create public members (data and functions).

The public members are accessible from any part of the program.

PowerPoint Slide Show - [C++ Technical Training CS] - PowerPoint

C++ Objects & Class

```
#include <iostream>
using namespace std;

// define a class
class Sample {
   // public elements
  public:
   int age;
   void displayAge() {
      cout << "Age = " << age << endl;
   }
};

int main() {
   // declare a class object
   Sample obj1;
   cout << "Enter your age: ";
  // store input in age of the obj1 object
   cin >> obj1.age;
   // call class function
   obj1.displayAge();
   return 0;
}
```

In this program, we have created a class named Sample, which contains a public variable age and a public function displayAge().

In main(), we have created an object of the Sample class named obj1. We then access the public elements directly by using the codes obj1.age and obj1.displayAge().

```cpp
//============================================================
/*
In this program, we have created a class named Sample, which
contains a public variable age and a public function displayAge().

In main(), we have created an object of the Sample class named
obj1. We then access the public elements directly by using the
codes obj1.age and obj1.displayAge().
*/

#include <iostream>
using namespace std;

// define a class
class Sample {
  // public elements
  public:
    int age;
    void displayAge(){
      cout << "Age = " << age << endl;
    }
};

int main(){
  // declare a class object
  Sample obj1;
  cout << "Enter your age: ";

  // store input in age of the obj1 object
  cin >> obj1.age;

  // call class function
  obj1.displayAge();

  return 0;
}

/*
Enter your age: 22
Age = 22

--------------------------------
Process exited after 2.809 seconds with return value 0
*/
```

PowerPoint Slide Show - [C++ Technical Training CS] - PowerPoint

C++ Objects & Class

private Access Modifier

The private keyword is used to create private members (data and functions).

The private members can only be accessed from within the class.

However, friend classes and friend functions can access private members.

ENG 7:42 PM

PowerPoint Slide Show - [C++ Technical Training CS] - PowerPoint

C++ Objects & Class

```
#include <iostream>
using namespace std;

// define a class
class Sample {
   // private elements
  private:
   int age;
   // public elements
  public:
   void displayAge(int a) {
      age = a;
      cout << "Age = " << age << endl;
   }
};

int main() {
   int ageInput;
   // declare an object
   Sample obj1;
   cout << "Enter your age: ";
   cin >> ageInput;
   // call function and pass ageInput as argument
   obj1.displayAge(ageInput);
   return 0;
}
```

In main(), the object obj1 cannot directly access the class variable age.

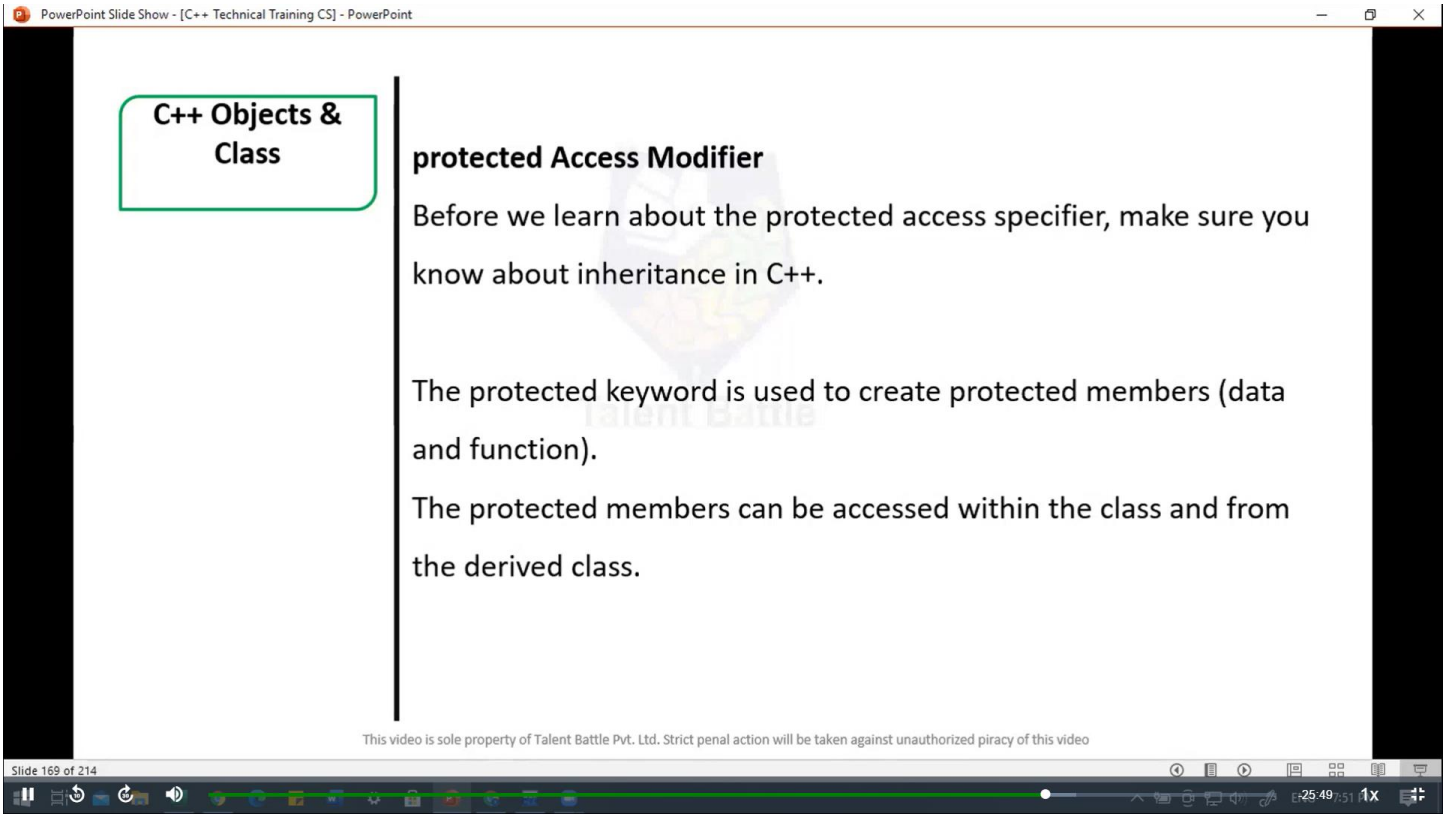
```
// error
cin >> obj1.age;
```

We can only indirectly manipulate age through the public function displayAge(), since this function initializes age with the value of the argument passed to it i.e. the function parameter int a.

33:37 1x

```
//================================================
/*
In main(), the object obj1 cannot directly acces the class
variable age.

// error
cin >> obj1.age;
We can only indirectly manipulate age through the public
funciton displayAge(), since this function initializes age
with the value of the argument passed to it i.e the funciton
parameer int a.
*/

#include <iostream>
using namespace std;

// define a class
class Sample {
  // private elements
  private:
    int age;

  // public elements
  public:
    void displayAge(int a){
      age = a;
      cout << "Age = " << age << endl;
    }
};

int main(){
  int ageInput;
  // declare an object
  Sample obj1;

  cout << "Enter you age: ";
  cin >> ageInput;

  // call function and pass ageInput as argument
  obj1.displayAge(ageInput);
  return 0;
}



/*
Enter you age: 22
Age = 22

--------------------------------
```

```
Process exited after 1.974 seconds with return value 0
*/
```



```cpp
//============================================
// Protected access modifier

#include <iostream>
using namespace std;

// declare parent class
class Sample {
  // protected elements
  protected:
    int age;
};

// declare child class
class SampleChild : public Sample {
  public:
    void displayAge(int a){
      age = a;
      cout << "Age = " << age << endl;
    }
};

int main(){
  int ageInput;
  // declare object of child class
  SampleChild child;
  cout << "Enter you age:  ";
  cin >> ageInput;

  // call child class function
  // pass ageInput as argument
  child.displayAge(ageInput);

  return 0;
}



/*
Enter you age:  22
Age = 22

--------------------------------
Process exited after 2.936 seconds with return value 0
*/
```

C++ Objects & Class

Here, SampleChild is an inherited class that is derived from Sample. The variable age is declared in Sample with the protected keyword.

This means that SampleChild can access age since Sample is its parent class.

We see this as we have assigned the value of age in SampleChild even though age is declared in the Sample class.

C++ Objects & Class

Summary: public, private, and protected

public elements can be accessed by all other classes and functions.

private elements cannot be accessed outside the class in which they are declared, except by friend classes and functions.

protected elements are just like the private, except they can be accessed by derived classes.

Specifiers	Same Class	Derived Class	Outside Class
public	Yes	Yes	Yes
private	Yes	No	No
protected	Yes	Yes	No

Note: By default, class members in C++ are private, unless specified otherwise.

C++ Objects & Class

C++ Constructors

A constructor is a special type of member function that is called automatically when an object is created.
In C++, a constructor has the same name as that of the class and it does not have a return type. For example,

```
class  Wall {
  public:
   // create a constructor
   Wall() {
      // code
   }
};
```

Here, the function Wall() is a constructor of the class Wall. Notice that the constructor

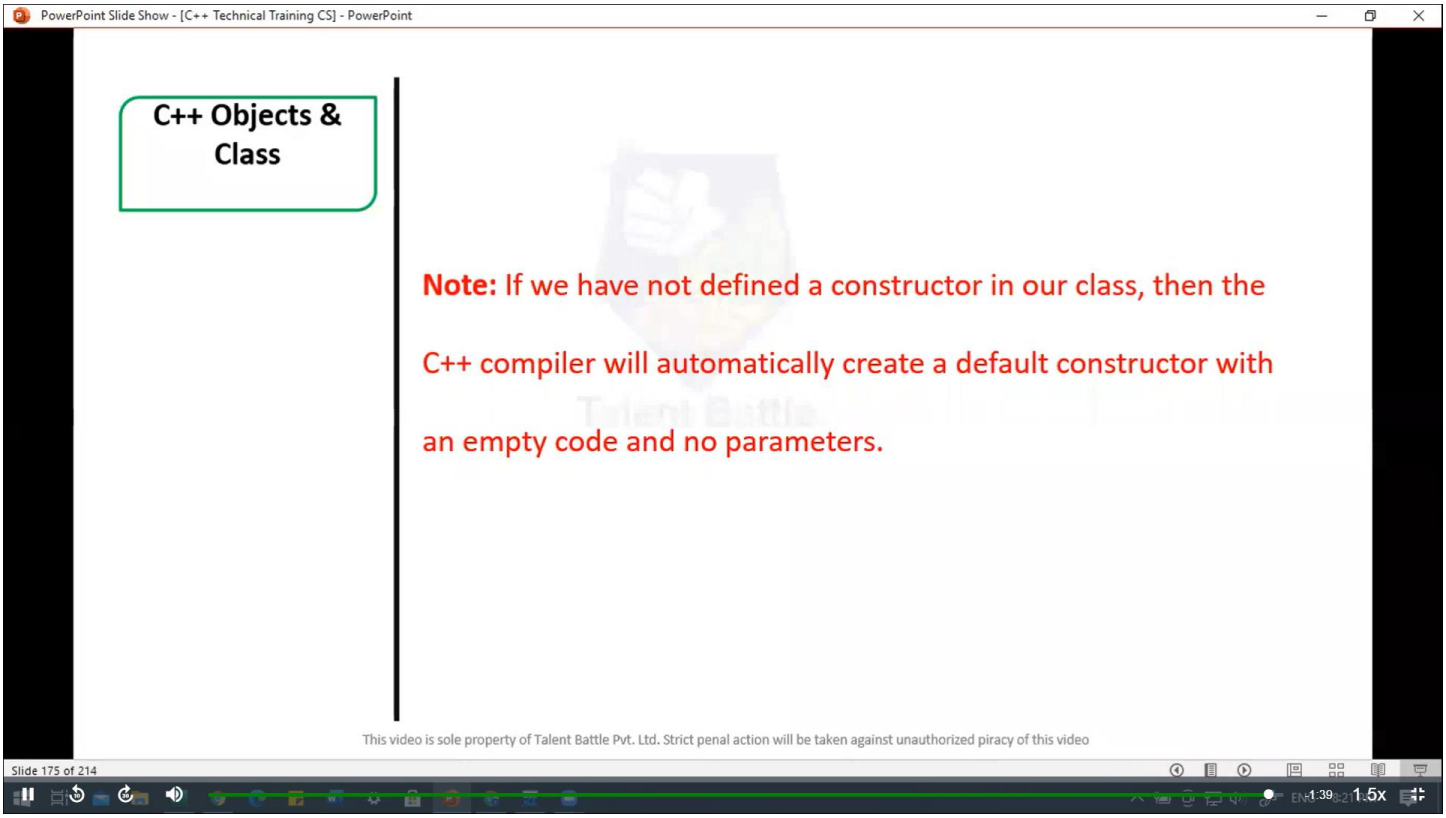
- has the same name as the class,
- does not have a return type, and
- is public

C++ Objects & Class

C++ Default Constructor

A constructor with no parameters is known as a default constructor.

```
// C++ program to demonstrate the use of default constructor

#include <iostream>
using namespace std;

// declare a class
class  Wall {

  private:
     double length;

  public:
   // create a constructor
   Wall() {

      // initialize private variables
      length = 5.5;

      cout << "Creating a wall." << endl;
      cout << "Length = " << length << endl;
   }
};

int main() {

   // create an object
   Wall wall1;

   return 0;
}
```


PowerPoint Slide Show - [C++ Technical Training CS] - PowerPoint
C++ Objects & Class
Note: If we have not defined a constructor in our class, then the C++ compiler will automatically create a default constructor with an empty code and no parameters.
This video is sole property of Talent Battle Pvt. Ltd. Strict penal action will be taken against unauthorized piracy of this video
Slide 175 of 214
1:39
1.5x

```cpp
// C++ Default Constructor
/*
A constructor with no parameters is known as a default constructor.
C++ program to demonstrate the use of default constructor
*/

#include <iostream>
using namespace std;

// declare a class
class Wall{
  private:
    double length;

  public:
    // create a constructor
    Wall(){
      // initialize private variables
      length = 5.5;

      cout << "Create a wall." << endl;
      cout << "Length = " << length << endl;
    }
};

int main(){
  // create an object
  Wall wall1;

  return 0;
}


/*
Create a wall.
Length = 5.5

--------------------------------
Process exited after 0.1771 seconds with return value 0
*/
```