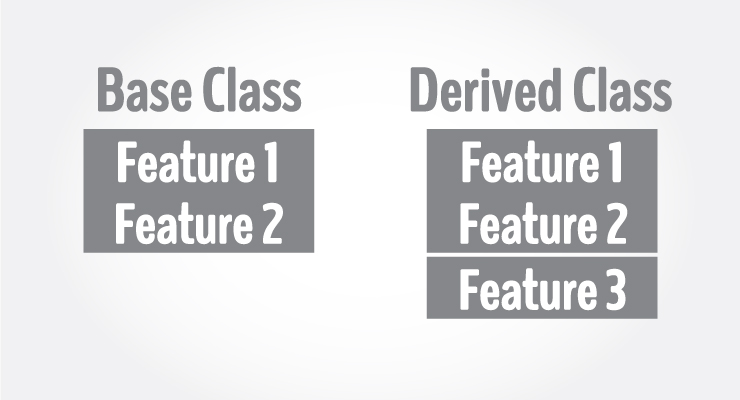
C++ Inheritance



Inheritance is one of the key features of Object-oriented programming in C++. It allows user to create a new class (derived class) from an existing class (base class).

The derived class inherits all the features from the base class and can have additional features of its own.

## Why inheritance should be used?

Suppose, in your game, you want three characters - a **maths teacher**, a **footballer** and a **businessman**.

Since, all of the characters are persons, they can walk and talk. However, they also have some special skills. A maths teacher can **teach maths**, a footballer can **play football** and a businessman can **run a business**.

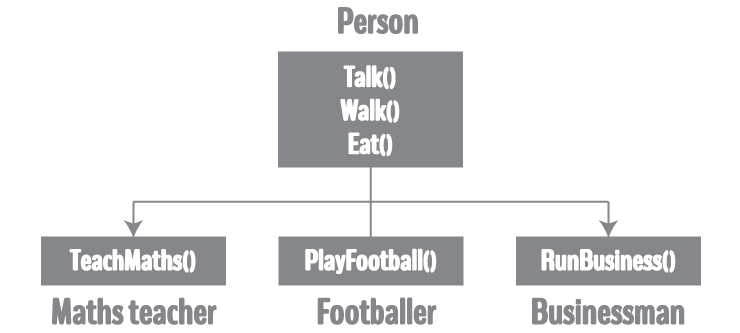
You can individually create three classes who can walk, talk and perform their special skill as shown in the figure below.



In each of the classes, you would be copying the same code for walk and talk for each character.

If you want to add a new feature - eat, you need to implement the same code for each character. This can easily become error prone (when copying) and duplicate codes.

It'd be a lot easier if we had a **Person** class with basic features like talk, walk, eat, sleep, and add special skills to those features as per our characters. This is done using inheritance.



Using inheritance, now you don't implement the same code for walk and talk for each class. You just need to **inherit** them.

So, for Maths teacher (derived class), you inherit all features of a Person (base class) and add a new feature **TeachMaths**. Likewise, for a footballer, you inherit all the features of a Person and add a new feature **PlayFootball** and so on.

This makes your code cleaner, understandable and extendable.

**It is important to remember:** When working with inheritance, each derived class should satisfy the condition whether it **"is a"** base class or not. In the example above, Maths teacher **is a** Person, Footballer **is a** Person. You cannot have: Businessman **is a** Business.

## Implementation of Inheritance in C++ Programming

class Person

{

... .. ...

};

class MathsTeacher : public Person

{

... .. ...

};

class Footballer : public Person

{

.... .. ...

};

In the above example, class Person is a base class and classes MathsTeacher and Footballerare the derived from *Person*.

The derived class appears with the declaration of a class followed by a colon, the keyword public and the name of base class from which it is derived.

Since, MathsTeacher and Footballer are derived from Person, all data member and member function of Person can be accessible from them.

## Example: Inheritance in C++ Programming

Create game characters using the concept of inheritance.

#include <iostream>

using namespace std;

class Person

{

public:

string profession;

int age;

Person(): profession("unemployed"), age(16) { }

void display()

{

cout << "My profession is: " << profession << endl;

cout << "My age is: " << age << endl;

walk();

talk();

}

void walk() { cout << "I can walk." << endl; }

void talk() { cout << "I can talk." << endl; }

};

// MathsTeacher class is derived from base class Person.

class MathsTeacher : public Person

{

public:

void teachMaths() { cout << "I can teach Maths." << endl; }

};

// Footballer class is derived from base class Person.

class Footballer : public Person

{

public:

void playFootball() { cout << "I can play Football." << endl; }

};

int main()

{

MathsTeacher teacher;

teacher.profession = "Teacher";

teacher.age = 23;

teacher.display();

teacher.teachMaths();

Footballer footballer;

footballer.profession = "Footballer";

footballer.age = 19;

footballer.display();

footballer.playFootball();

return 0;

}

**Output**

My profession is: Teacher

My age is: 23

I can walk.

I can talk.

I can teach Maths.

My profession is: Footballer

My age is: 19

I can walk.

I can talk.

I can play Football.

In this program, Person is a base class, while MathsTeacher and Footballer are derived from Person.

Person class has two data members - profession and age. It also has two member functions - walk() and talk().

Both MathsTeacher and Footballer can access all data members and member functions of Person.

However, MathsTeacher and Footballer have their own member functions as well: teachMaths() and playFootball() respectively. These functions are only accessed by their own class.

In the main() function, a new MathsTeacher object teacher is created.

Since, it has access to Person's data members, profession and age of teacher is set. This data is displayed using the display() function defined in the Person class. Also, the teachMaths() function is called, defined in the MathsTeacher class.

Likewise, a new Footballer object footballer is also created. It has access to Person's data members as well, which is displayed by invoking the display() function. The playFootball()function only accessible by the footballer is called then after.

### Access specifiers in Inheritance

When creating a derived class from a base class, you can use different access specifiers to inherit the data members of the base class.

These can be public, protected or private.

In the above example, the base class Person has been inherited public-ly by MathsTeacherand Footballer.

Learn more about Public, Protected and Private Inheritance in C++.

### Member Function Overriding in Inheritance

Suppose, base class and derived class have member functions with same name and arguments.

If you create an object of the derived class and try to access that member function, the member function in derived class is only invoked.

The member function of derived class overrides the member function of base class.