**Why inheritance should be used?**

Suppose, you want to create 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**.

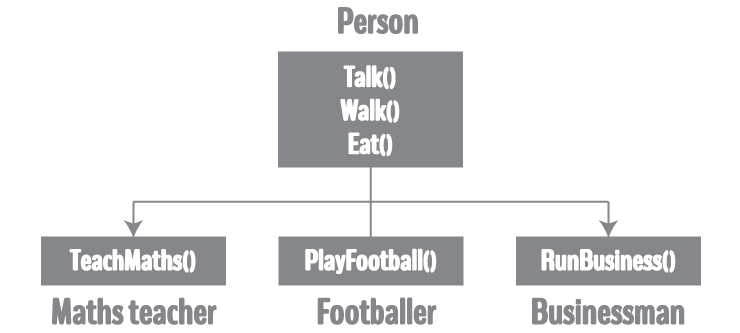
You can individually create three classes who can walk, talk and perform their special skill as shown in the figure below.



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 Footballer are 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();

}

**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.

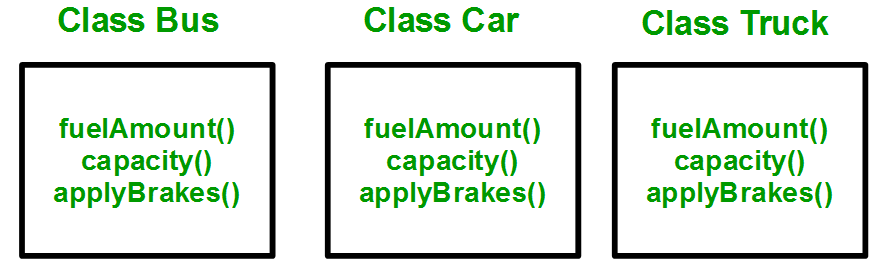
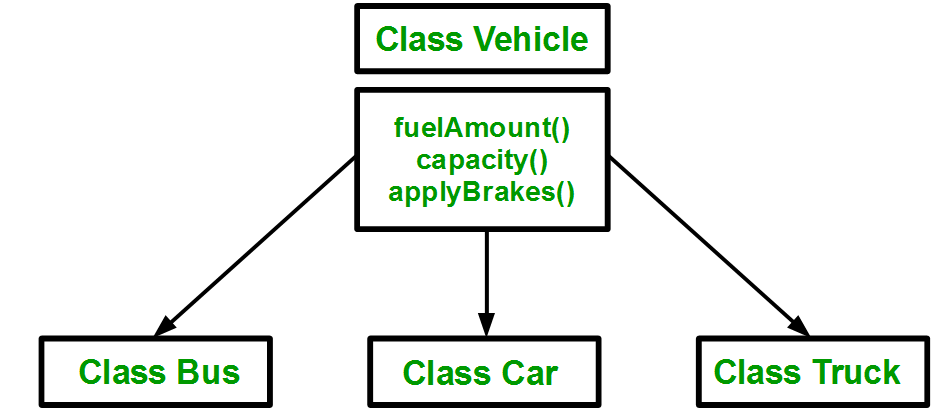
However, MathsTeacher and Footballer have their own member functions as well: teachMaths() and playFootball() respectively. These functions are only accessed by their own class.

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.

**Why and when to use inheritance?**

Consider a group of vehicles. You need to create classes for Bus, Car and Truck. The methods fuelAmount(), capacity(), applyBrakes() will be same for all of the three classes. If we create these classes avoiding inheritance then we have to write all of these functions in each of the three classes as shown in below figure:  
[](http://cdncontribute.geeksforgeeks.org/wp-content/uploads/inheritance.png)  
You can clearly see that above process results in duplication of same code 3 times. This increases the chances of error and data redundancy. To avoid this type of situation, inheritance is used. If we create a class Vehicle and write these three functions in it and inherit the rest of the classes from the vehicle class, then we can simply avoid the duplication of data and increas re-usability. Look at the below diagram in which the three classes are inherited from vehicle class:[](http://cdncontribute.geeksforgeeks.org/wp-content/uploads/inheritance2.png)  
Using inheritance, we have to write the functions only one time instead of three times as we have inherited rest of the three classes from base class(Vehicle).

**Implementing inheritance in C++**: For creating a sub-class which is inherited from the base class we have to follow the below syntax.  
**Syntax**:

class subclass\_name : access\_mode base\_class\_name

{

//body of subclass

};

Here, **subclass\_name**is the name of the sub class, **access\_mode**is the mode in which you want to inherit this sub class for example: public, private etc. and **base\_class\_name**is the name of the base class from which you want to inherit the sub class.  
**Note**: private member of the base class will never get inherited in the sub class.

|  |
| --- |
| // C++ program to demonstrate implementation  // of Inheritance    #include <bits/stdc++.h>  using namespace std;    //Base class  class Parent  {      public:        int id\_p;  };    // Sub class inheriting from Base Class(Parent)  class Child : public Parent  {      public:        int id\_c;  };    //main function  int main()     {            Child obj1;            // An object of class child has all data members          // and member functions of class parent          obj1.id\_c = 7;          obj1.id\_p = 91;          cout << "Child id is " <<  obj1.id\_c << endl;          cout << "Parent id is " <<  obj1.id\_p << endl;            return 0;     } |

Run on IDE

Output:

Child id is 7

Parent id is 91

In the above program the ‘Child’ class is publicly inherited from the ‘Parent’ class so the public data members of the class ‘Parent’ will also be inherited by the class ‘Child’.

// C++ program to explain

// Single inheritance

#include <iostream>

using namespace std;

// base class

class Vehicle {

  public:

    Vehicle()

    {

      cout << "This is a Vehicle" << endl;

    }

};

// sub class derived from two base classes

class Car: public Vehicle{

};

// main function

int main()

{

    // creating object of sub class will

    // invoke the constructor of base classes

    Car obj;

    return 0;

}