

Inheritance in C++

Inheritance is one of the feature of Object Oriented Programming System(OOPs), it allows the child class to acquire the properties (the data members) and functionality (the member functions) of parent class.

What is child class?

A class that inherits another class is known as child class, it is also known as derived class or subclass.

What is parent class?

The class that is being inherited by other class is known as parent class, super class or base class.

Syntax of Inheritance

```
class parent_class
{
    //Body of parent class
};
class child_class : access_modifier parent_class
{
    //Body of child class
};
```

What are the advantages of using inheritance in C++ Programming

The main advantages of inheritance are **code reusability** and **readability**. When child class inherits the properties and functionality of parent class, we need not to write the same code again in child class. This makes it easier to reuse the code, makes us write the less code and the code becomes much more readable.

Lets take a **real life example** to understand this: Lets assume that **Human** is a class that has properties such as height, weight, colour etc and functionality such as eating(), sleeping(), dreaming(), working() etc.

Now we want to create **Male** and **Female** class, these classes are different but since both Male and Female are humans they share some common properties and behaviours (functionality) so they can inherit those properties and functionality from Human class and rest can be written in their class separately.

This approach makes us write less code as both the classes inherited several properties and functions from base class thus we didn't need to re-write them. Also, this makes it easier to read the code.

Inheritance Example

Before we discuss the types of inheritance, let's take an example:

Here we have two classes `Teacher` and `MathTeacher`, the `MathTeacher` class inherits the `Teacher` class which means `Teacher` is a parent class and `MathTeacher` is a child class. The child class can use the property `collegeName` of parent class.

Another important point to note is that when we create the object of child class it calls the constructor of child class and child class constructor automatically calls the constructor of base class.

```
#include <iostream>
using namespace std;
class Teacher {
public:
    Teacher(){
        cout<<"Hey Guys, I am a teacher"<<endl;
    }
    string collegeName = "Beginnersbook";
};
//This class inherits Teacher class
class MathTeacher: public Teacher {
public:
    MathTeacher(){
        cout<<"I am a Math Teacher"<<endl;
    }
    string mainSub = "Math";
    string name = "Negan";
};
int main() {
    MathTeacher obj;
    cout<<"Name: "<<obj.name<<endl;
    cout<<"College Name: "<<obj.collegeName<<endl;
    cout<<"Main Subject: "<<obj.mainSub<<endl;
    return 0;
}
```

Output:

```
Hey Guys, I am a teacher
I am a Math Teacher
Name: Negan
College Name: Beginnersbook
Main Subject: Math
```

Types of Inheritance in C++

- 1) Single inheritance
- 2) Multilevel inheritance
- 3) Multiple inheritance
- 4) Hierarchical inheritance
- 5) Hybrid inheritance

Single inheritance

In Single inheritance one class inherits one class exactly.
For example: Let's say we have class A and B

B inherits A

Example of Single inheritance:

```
#include <iostream>
using namespace std;
class A {
public:
    A(){
        cout<<"Constructor of A class"<<endl;
    }
};
class B: public A {
public:
    B(){
        cout<<"Constructor of B class";
    }
};
int main() {
    //Creating object of class B
    B obj;
    return 0;
}
```

Output:

Constructor of A class
Constructor of B class

2)Multilevel Inheritance

In this type of inheritance one class inherits another child class.

C inherits B and B inherits A

Example of Multilevel inheritance:

```
#include <iostream>
using namespace std;
class A {
public:
    A(){
        cout<<"Constructor of A class"<<endl;
    }
};
class B: public A {
public:
    B(){
        cout<<"Constructor of B class"<<endl;
    }
};
class C: public B {
public:
    C(){
        cout<<"Constructor of C class"<<endl;
    }
};
int main() {
    //Creating object of class C
}
```

```
C obj;
return 0;
}
```

Output:

```
Constructor of A class
Constructor of B class
Constructor of C class
```

Multiple Inheritance

In multiple inheritance, a class can inherit more than one class. This means that in this type of inheritance a single child class can have multiple parent classes.

For example:

C inherits A and B both

Example of Multiple Inheritance:

```
#include <iostream>
using namespace std;
class A {
public:
    A(){
        cout<<"Constructor of A class"<<endl;
    }
};
class B {
public:
    B(){
        cout<<"Constructor of B class"<<endl;
    }
};
class C: public A, public B {
public:
    C(){
        cout<<"Constructor of C class"<<endl;
    }
};
int main() {
    //Creating object of class C
    C obj;
    return 0;
}
Constructor of A class
Constructor of B class
Constructor of C class
```

4) Hierarchical Inheritance

In this type of inheritance, one parent class has more than one child class. For example:

Class B and C inherits class A

Example of Hierarchical inheritance:

```
#include <iostream>
```

```
using namespace std;
class A {
public:
    A(){
        cout<<"Constructor of A class"<<endl;
    }
};
class B: public A {
public:
    B(){
        cout<<"Constructor of B class"<<endl;
    }
};
class C: public A{
public:
    C(){
        cout<<"Constructor of C class"<<endl;
    }
};
int main() {
    //Creating object of class C
    C obj;
    return 0;
}
```

Output:

```
Constructor of A class
Constructor of C class
```

5) Hybrid Inheritance

Hybrid inheritance is a combination of more than one type of inheritance. For example, A child and parent class relationship that follows multiple and hierarchical inheritance both can be called hybrid inheritance.