Virtual function vs Pure virtual function in C++

Before understanding the differences between the virtual function and pure virtual function in C++, we should know about the virtual function and pure virtual function in C++.

What is virtual function?

Virtual function is a member function that is declared within the base class and can be redefined by the derived class.

Let's understand through an example.

```
#include <iostream>
using namespace std;
class base
{
    public:
    void show()
    {
        std::cout << "Base class" << std::endl;
    }
};
class derived1 : public base
{
    public:
    void show()
    {
        std::cout << "Derived class 1" << std::endl;
    }
};
class derived2 : public base
{
    public:
    void show()
    {
        std::cout << "Derived class 2" << std::endl;
    }
};
int main()
{
    base *b;
    derived1 d1;
    derived2 d2;
    b=&d1;
    b->show();
    b=&d2;
    b->show();
    return 0;
}
```

In the above code, we have not used the virtual method. We have created a base class that contains the show() function. The two classes are also created named as '**derived1**' and '**derived2**' that are inheriting the properties of the base class. Both 'derived1' and 'derived2' classes have redefined the show() function. Inside the main() method, pointer variable 'b' of class base is declared. The objects of classes derived1 and derived2 are d1 and d2 respectively. Although the 'b' contains the addresses of d1 and d2, but when calling the show() method; it always calls the show() method of the base class rather than calling the functions of the derived1 and derived2 class.

To overcome the above problem, we need to make the method as virtual in the base class. Here, virtual means that the method exists in appearance but not in reality. We can make the method as virtual by simply adding the virtual keyword preceeding to the function. In the above program, we need to add the virtual keyword that precedes to the show() function in the base class shown as below:

```
virtual void show()
{
    std::cout << "Base class" << std::endl;
}
```

Once the above changes are made, the output would be:

Important points:

- It is a run-time polymorphism.

- Both the base class and the derived class have the same function name, and the base class is assigned with an address of the derived class object then also pointer will execute the base class function.
- If the function is made virtual, then the compiler will determine which function is to execute at the run time on the basis of the assigned address to the pointer of the base class.

What is pure virtual function?

A pure virtual function is a virtual function that has no definition within the class. Let's understand the concept of pure virtual function through an example.

In the above pictorial representation, shape is the base class while rectangle, square and circle are the derived class. Since we are not providing any definition to the virtual function, so it will automatically be converted into a pure virtual function.

Characteristics of a pure virtual function

- A pure virtual function is a "do nothing" function. Here "do nothing" means that it just provides the template, and derived class implements the function.
- It can be considered as an empty function means that the pure virtual function does not have any definition relative to the base class.
- Programmers need to redefine the pure virtual function in the derived class as it has no definition in the base class.
- A class having pure virtual function cannot be used to create direct objects of its own. It means that the class is containing any pure virtual function then we cannot create the object of that class. This type of class is known as an abstract class.

Syntax

There are two ways of creating a virtual function:

```
virtual void display() = 0;
```

or

```
virtual void display() {}
```

Differences between the virtual function and pure virtual function

Virtual function	**Pure virtual function**
A virtual function is a member function in a base class that can be redefined in a derived class.	A pure virtual function is a member function in a base class whose declaration is provided in a base class and implemented in a derived class.
The classes which are containing virtual functions are not abstract classes.	The classes which are containing pure virtual function are the abstract classes.
In case of a virtual function, definition of a function is provided in the base class.	In case of a pure virtual function, definition of a function is not provided in the base class.
The base class that contains a virtual function can be instantiated.	The base class that contains a pure virtual function becomes an abstract class, and that cannot be instantiated.
If the derived class will not redefine the virtual function of the base class, then there will be no effect on the compilation.	If the derived class does not define the pure virtual function; it will not throw any error but the derived class becomes an abstract class.
All the derived classes may or may not redefine the virtual function.	All the derived classes must define the pure virtual function.