VIRTUAL FUNCTION IN C++

VIRTUAL FUNCTION IN C++

- A virtual function is a member function which is declared within a base class and is re-defined(Overridden) by a derived class.
- When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class's version of the function.
- Virtual functions ensure that the correct function is called for an object, regardless of the type of reference (or pointer) used for function call.
- They are mainly used to achieve Runtime polymorphism
- Functions are declared with a virtual keyword in base class.
- The resolving of function call is done at Run-time.

- Virtual functions cannot be static.
- A virtual function can be a friend function of another class.
- Virtual functions should be accessed using pointer or reference of base class type to achieve run time polymorphism.
- The prototype of virtual functions should be the same in the base as well as derived class.
- They are always defined in the base class and overridden in a derived class. It is not mandatory for the derived class to override (or re-define the virtual function), in that case, the base class version of the function is used.
- A class may have virtual destructor but it cannot have a virtual constructor.

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

class base {
public:
    virtual void print()
    {
    cout << "print base class" << endl;
    }

    void show()
    {
    cout << "show base class" << endl;
    }
};
```

```cpp
class derived : public base
{
public:
void print()
{
cout << "print derived class" << endl;
}

void show()
{
cout << "show derived class" << endl;
}
};
```

```
int main()
{
   base* bptr;
   derived d;
   bptr = &d;

   // virtual function, binded at runtime
   bptr->print();

   // Non-virtual function, binded at compile time
   bptr->show();
}
```