# Can virtual functions be private in C++?

Difficulty Level : Medium • Last Updated : 23 Aug, 2021

In C++, virtual functions can be private and can be overridden by the derived class. For example, the following program compiles and runs fine.

**C++**

```
#include <iostream>

class base
{
public:

    // default base constructor
    base() { std::cout << "base class constructor\n"; }

    // virtual base destructor
    // always use virtual base
    // destructors when you know you
    // will inherit this class
    virtual ~base()
    {
        std::cout << "base class destructor\n";
    }
    // public method in base class
    void show()
    {
        std::cout << "show() called on base class\n";
    }

    // public virtual function in base class,
    // contents of this function are printed when called
    // with base class object when called with base class
    // pointer contents of derived class are printed on
    // screen
    virtual void print()
    {
        std::cout << "print() called on base class\n";
    }
};

class derived : public base {
public:
    // default derived constructor
    derived()
        : base()
    {
        std::cout << "derived class constructor\n";
    }
    // virtual derived destructor
    // always use virtual destructors
    // when inheriting from a
    // base class
    virtual ~derived()
    {
        std::cout << "derived class destructor\n";
    }

private:
    // private virtual function in derived class overwrites
    // base class virtual method contents of this function
    // are printed when called with base class pointer or
    // when called with derived class object
    virtual void print()
    {
        std::cout << "print() called on derived class\n";
    }
};

int main()
{
    std::cout << "printing with base class pointer\n";

    // construct base class pointer with derived class
    // memory
    base* b_ptr = new derived();

    // call base class show()
    b_ptr->show();

    // call virtual print in base class but it is overridden
    // in derived class also note that print() is private
    // member in derived class, still the contents of derived
    // class are printed this code works because base class
    // defines a public interface and drived class overrides
    // it in its implementation
    b_ptr->print();

    delete b_ptr;
}
```

**Output**

```
printing with base class pointer
base class constructor
derived class constructor
show() called on base class
print() called on derived class
derived class destructor
base class destructor
```

There are few things to note in the above program.
b_ptr is a pointer of Base type and points to a Derived class object. When ptr->print() is called, print() of Derived is executed.

This code works because base class defines a public interface and derived class overrides it in its implementation even though derived has a private virtual function.
Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Want to learn from the best curated videos and practice problems, check out the C++ Foundation Course for Basic to Advanced C++ and C++ STL Course for foundation plus STL. To complete your preparation from learning a language to DS Algo and many more, please refer **Complete Interview Preparation Course**.

Like 30