Object-Oriented Programming (OOP)

Week – 09

April 06-10, 2020

Instructor: **Basit Ali**

Email: basit.jasani@nu.edu.pk

Object-Oriented Programming (OOP)

Virtual Functions

- A virtual function a member function which is declared within base class and is re-defined (Overridden) by 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.

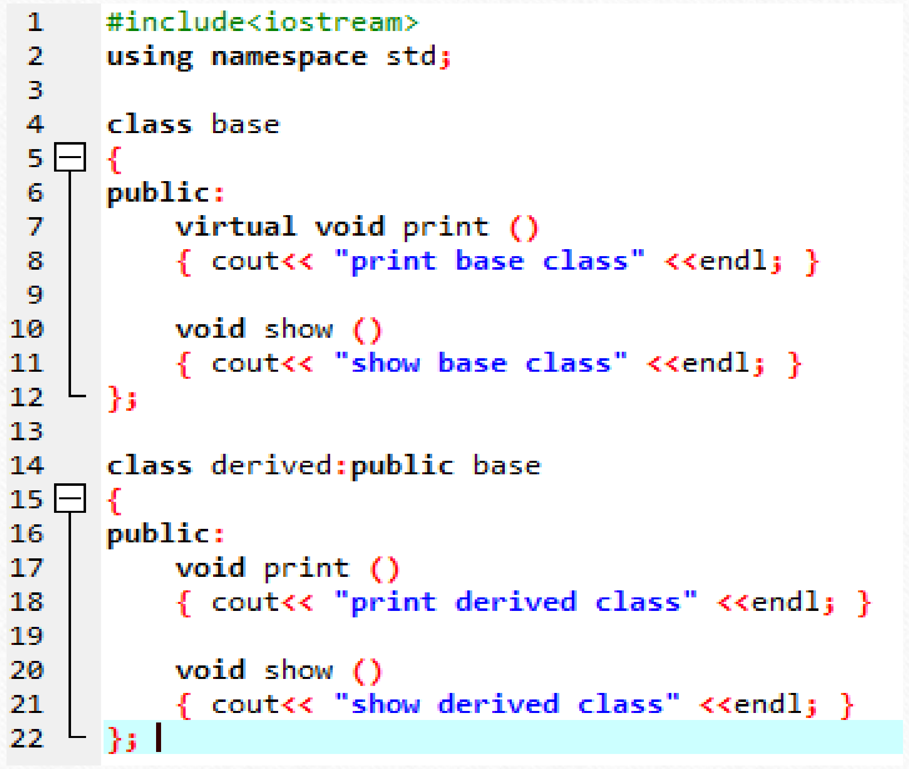
Virtual Functions

- 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.

Rules for Virtual Functions

- They Must be declared in public section of class.
- Virtual functions cannot be static and also cannot be a friend function of another class.
- They are always defined in base class and overridden in derived class. It is not mandatory for derived class to override (or re-define the virtual function), in that case base class version of function is used.

Compile-time(early binding) VS run-time(late binding) behavior of Virtual Functions



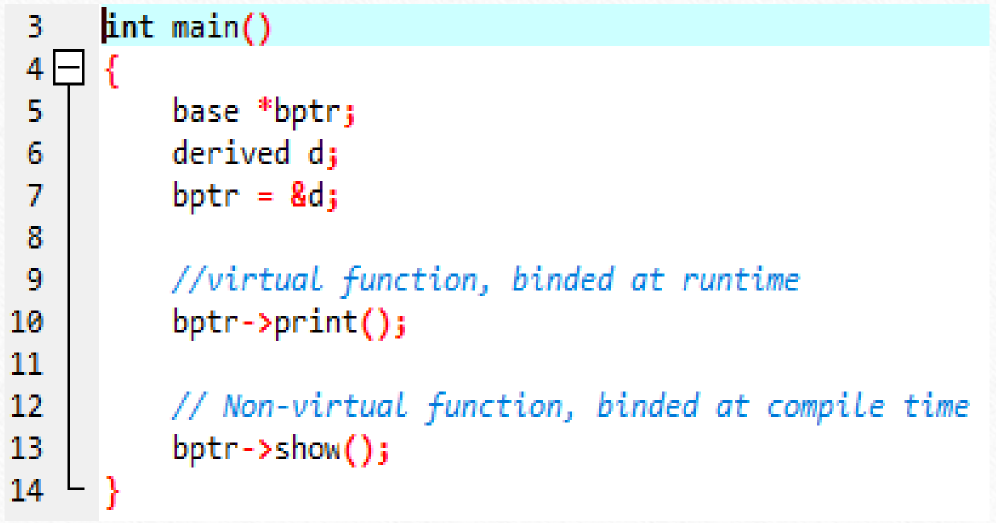
```
#include<iostream>
using namespace std;

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

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

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();
}
```

print derived class
show base class

Pure Virtual Function

- A pure virtual function is specified by placing **“= 0”** in its declaration

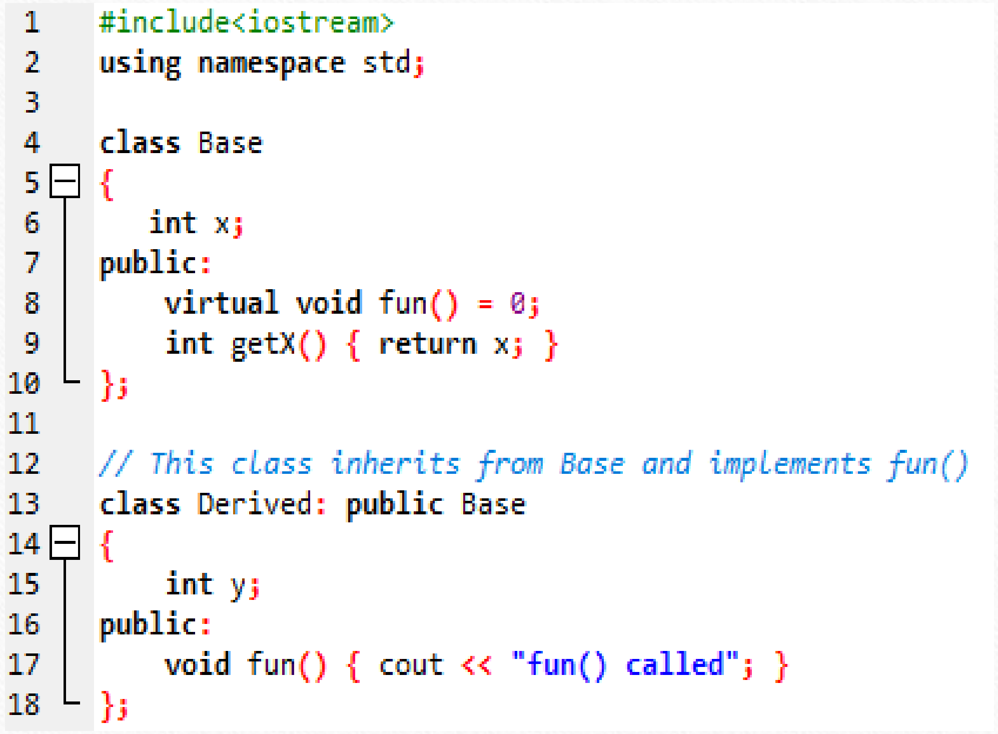
Example: virtual void draw() = 0;

Virtual vs Pure Virtual

- A virtual function has an implementation in the base class; a pure virtual function does not have an implementation in the base class.

- Virtual functions **can be** overridden by the derived classes; pure virtual functions **must be** overridden by the derived classes.

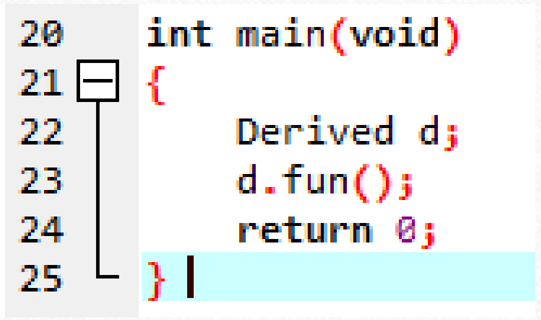
Example



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

class Base
{
   int x;
public:
    virtual void fun() = 0;
    int getX() { return x; }
};

// This class inherits from Base and implements fun()
class Derived: public Base
{
    int y;
public:
    void fun() { cout << "fun() called"; }
};
```



```
int main(void)
{
    Derived d;
    d.fun();
    return 0;
}
```

Abstract Classes

- An **abstract class** is a base class that will never have an object instantiated from it.
  - **Abstract classes** are used only for inheritance, they are too general to create objects from.
  - **Abstract classes** provide an generic base class that can be used by concrete classes to provide an interface and/or implementation.

Abstract Classes

- An abstract class in C++ is defined as any class that contains at least one **pure virtual function**.

Concepts!

1) A class is abstract if it has at least one pure virtual function.

2) We can have pointers and references of abstract class type.

3) If we do not override the pure virtual function in derived class, then derived class also becomes abstract class.

4) An abstract class can have constructors.

Interface vs Abstract Classes

- An interface does not have implementation of any of its methods, it can be considered as a collection of method declarations
- In C++, an interface can be simulated by making all methods as pure virtual.
- In Java, there is a separate keyword for interface.

Case Study!

A company pays its employees weekly.

The employees are of four types: Salaried employees are paid a fixed weekly salary regardless of the number of hours worked, hourly employees are paid by the hour and receive overtime pay for all hours worked in excess of 40 hours, commission employees are paid a percentage of their sales and base-salary-plus-commission employees receive a base salary plus a % of their sales. For the current pay period, the company has decided to reward base-salary-plus-commission employees by adding 10 percent to their base salaries.