LECTURE-10

OBJECT ORIENTED PROGRAMMING C++

Dr. Farhana Sarker
Assistant Professor
CSE, ULAB

22/09/2020

VIRTUAL FUNCTION

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

Virtual functions are declared with a **virtual** keyword in base class.

It used to achieve Runtime Polymorphism.

RULES OF VIRTUAL FUNCTION

- Virtual functions cannot 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 same in base as well as derived 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.
- A class may have virtual destructor but it cannot have a virtual constructor.

EXAMPLE OF VIRTUAL FUNCTION

```
#include <iostream>

using namespace std;

class A {

public:

virtual void print()

            {  cout << "print base class" << endl;   }

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

};
```

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

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

EXAMPLE OF VIRTUAL FUNCTION (CONT.)

```
int main()
{
    A *bp, obj;
    B d;
bp = &d;
       bp->print();
      bp->show();

bp = &obj;
       bp->print();
      bp->show();

}
```

PURE VIRTUAL FUNCTION

- A virtual function whose declaration ends with = 0 is called a pure virtual function.
- A class containing pure virtual function is known as abstract class.
- we cannot create object from abstract classes.
- *If we do not override the pure virtual function in derived class, then derived class also becomes abstract class.*

EXAMPLE OF PURE VIRTUAL FUNCTION

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

class Base
{
public:
    virtual void show() = 0;
    void print(){cout<<“PARENT”;}
};

class Derived: public Base
{
public:
    void show() { cout << "In Derived \n"; }
};
```

```
int main(void)
{
    Base *bp;
    Derived d;
    bp=&d;
    bp->show();

    bp->print();
    return 0;

}
```