BCA SEMESTER-II

Object Oriented Programming using C++

Paper Code: BCA CC203

Unit :4

Virtual Function in C++

By: Ms. Nimisha Manan

Assistant Professor

Dept. of Computer Science

Patna Women's College

Virtual Function

When the same function name is defined in the base class as well as the derived class , then the function in the base class is declared as Virtual Function. Thus a Virtual function can be defined as " A special member function that is declared within a base class and redefined by a derived class is known as virtual function". To declare a virtual function, in the base class precede the function prototype with the keyword virtual.

Syntax for defining a virtual function is as follows:-

```
virtual return_type function_name(arguments)
{
        ------
 }
```

Through virtual function, functions of the base class can be overridden by the functions of the derived class. With the help of virtual function, runtime polymorphism or dynamic polymorphism which is also known as late binding is implemented on that function. A function call is resolved at runtime in late binding and so compiler determines the type of object at runtime. Late binding allows binding between the function call and the appropriate virtual function (to be called) to be done at the time of execution of the program.

A class that declares or inherits a virtual function is called a polymorphic class.

For Example:

```
Virtual void show()
  {
      Cout<<"This is a virtual function";
    }
```

Implementation of Dynamic Polymorphism through Virtual Function

Dynamic Polymorphism is implemented through virtual function by using a single pointer to the base class that points to all the objects of derived class classes. At the time of execution, the appropriate function is called depending on the object which is currently being pointed by the base pointer variable. Thus by making the base pointer point to different objects, different versions of virtual function can be executed.

Rules for Virtual Functions

1. A Virtual Function must be defined in the public section of the base class.
2. The prototype of virtual functions should be same in base as well as derived class
3. Virtual function can not be a friend to another class.
4. Virtual functions cannot be static
5. Virtual functions should be accessed using pointer to base class which has memory address of the derived class object to achieve run time polymorphism.

Example :

```
#include <iostream.h>

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, bound at runtime
    bptr->print();

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

Output:

```
print derived class

show base class
```

Explanation: Runtime polymorphism is achieved only through a pointer (or reference) of base class type. Also, a base class pointer can point to the objects of base class as well as to the objects of derived class. In above code, base class pointer ‘bptr’ contains the address of object ‘d’ of derived class.
Late binding(Runtime) is done in accordance with the content of pointer (i.e. location pointed to by pointer) and Early binding(Compile time) is done according to the type of pointer, since print() function is declared with virtual keyword so it will be bound at run-time (output is *print derived class* as pointer is pointing to object of derived class) and show() is non-virtual so it will be bound during compile time(output is *show base class* as pointer is of base type).
NOTE: If we have created a virtual function in the base class and it is being overridden in the derived class then we don’t need virtual keyword in the derived class, functions are automatically considered as virtual functions in the derived class.