Lab Sheet: Exploring Virtual Functions in C++

Objective:

To understand and implement virtual functions in C++ to achieve runtime polymorphism. You will learn how to use virtual functions to call derived class methods through base class pointers.

Overview:

Virtual functions in C++ allow derived classes to override methods defined in base classes.

When a method is declared as `virtual`, C++ uses dynamic binding to determine which method to call at runtime based on the actual object type.

A virtual function in C++ is a member function in a base class that you can override in a derived class. It allows runtime polymorphism, meaning the appropriate function is called for an object, depending on its actual derived type, even if accessed via a base class pointer.

Syntax:

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

class Base {
public:
   virtual void show() {  // virtual function
      cout << "Base class show() called" << endl;
   }
};

class Derived : public Base {
public:
   void show() override {  // override keyword is optional but recommended
      cout << "Derived class show() called" << endl;
   }
};

int main() {
   Base* basePtr;         // base class pointer
   Derived d;
   basePtr = &d;

   basePtr->show();       // calls Derived's show() due to virtual function
   return 0;
}
```

Virtual Destructor

A virtual destructor ensures that when a base class pointer deletes a derived class object, both the base and derived class destructors are called properly.

Syntax:

```
class Base {
public:
   virtual   ~Base();     //   Virtual
destructor
};
```

Explanation:

- `virtual` keyword ensures polymorphic destruction.
- `~Base();` is the destructor of the base class.
- Must be declared `virtual` if you expect to delete derived objects through base pointers

Pure Virtual Functions and Abstract Classes

In C++, a pure virtual function is a special type of virtual function that has no definition in the base class and is specified by assigning = 0 in its declaration. It acts as a placeholder to be overridden by derived classes, enforcing a contract that they must provide their own implementation of that function. A class containing at least one pure virtual function is called an abstract class. Abstract classes serve as base classes and cannot be instantiated directly.

Syntax:

```
class AbstractClass {
public:
   virtual void pureVirtualFunction() = 0;  // Pure virtual function
};
```

Virtual Functions with Multiple Inheritance

In multiple inheritance, a class inherits from more than one base class. If both base classes have virtual functions with the same name, and the derived class overrides them, the correct function is

called at runtime based on the actual object type. Virtual functions resolve ambiguity and enable runtime polymorphism in multiple inheritance.

Syntax:

```
class A {
public:
    virtual void display();
};

class B {
public:
    virtual void display();
};

class C : public A, public B {
public:
    void display() override;
};
```