# VIRTUAL FUNCTION

## AIM:

To create a virtual function to demonstrate the runtime polymorphism.

## ALGORITHM:

· Define a base class Employee with protected data members Name and EmployeeID.

· Create a parameterized constructor in Employee to initialize employee details.

· Declare a virtual function display() in the base class to show employee information.

· Define a derived class Salaried inheriting from Employee.

· Add data members for working days and daily wage in the Salaried class.

· Create a constructor in Salaried to initialize salary-related data.

· Override the display() function in Salaried to calculate and print total salary.

· Define another derived class ContractLabour inheriting from Employee.

· Add data members for days worked and daily wage in ContractLabour.

· Create a constructor in ContractLabour to initialize wage details.

· Override the display() function in ContractLabour to calculate and print total wages.

· In the main() function, create base class pointers for derived class objects.

· Assign a Salaried object to the first base class pointer.

· Assign a ContractLabour object to the second base class pointer.

· Call the display() function using the base class pointers.

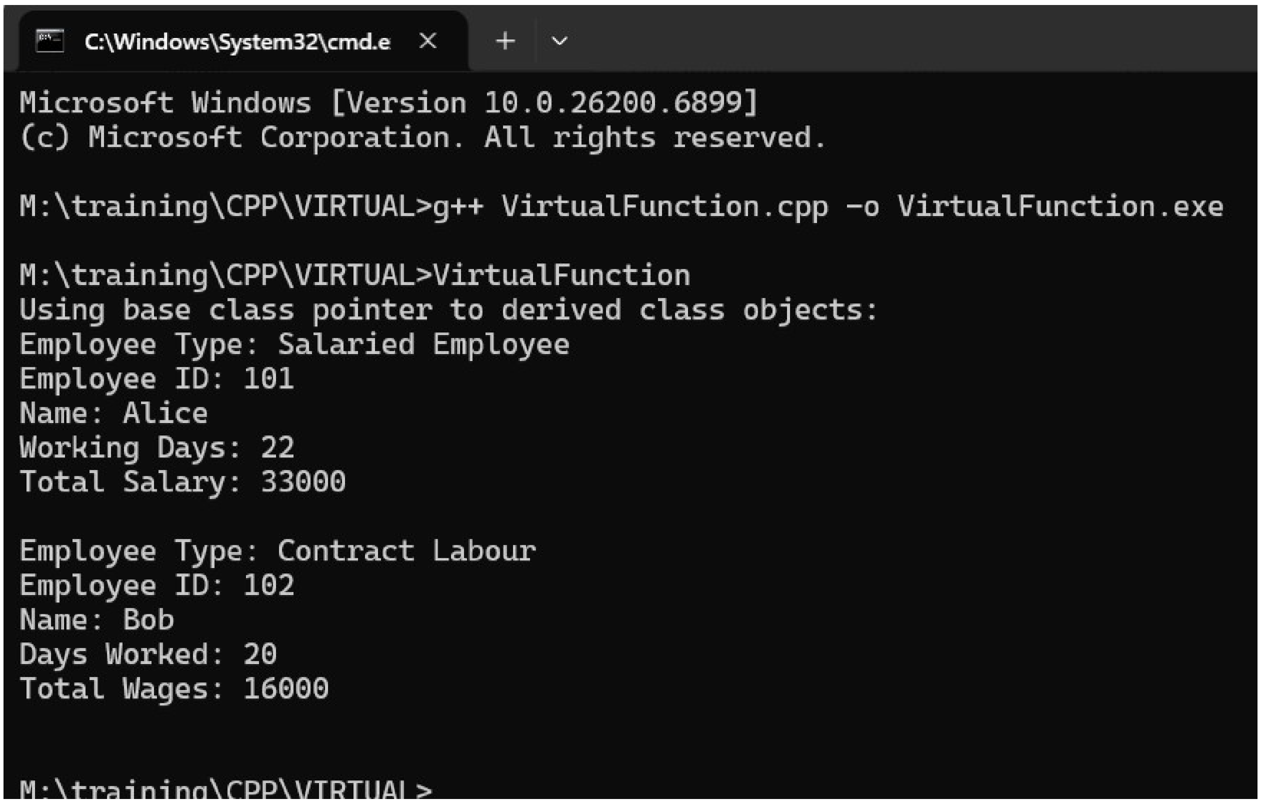
· Observe that the derived class display() functions are executed due to virtual functions.

## PROGRAM:

```
/*
*   Program to demonstrate virtual functions in C++
*   Author  : MUTHUGANESH S
*   Date    : 28/1/2026
*   Filename: VirtualFunction.cpp
*   retval  : void
*/

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

```cpp
class Employee{
protected:
    string Name;
    int EmployeeID;
public:
    Employee(int id, string name) : EmployeeID(id), Name(name) {}

    // Virtual function to display employee details
    virtual void display(){
        cout << "Employee Details" << endl;
        cout << "Name: " << Name << ", Employee ID: " << EmployeeID << endl;
    }
};

class Salaried: public Employee{
    int WorkingDays;
    int DailyWage;
public:
    Salaried(int id, string n, int days, float salary)
        : Employee(id, n) {
        WorkingDays = days;
        DailyWage = salary;
    }

    // Overriding display function
    void display() {
        float totalSalary = WorkingDays * DailyWage;
        cout << "Employee Type: Salaried Employee" << endl;
        cout << "Employee ID: " << EmployeeID << endl;
        cout << "Name: " << Name << endl;
        cout << "Working Days: " << WorkingDays << endl;
        cout << "Total Salary: " << totalSalary << endl<<endl;
    }
};

class ContractLabour: public Employee{
    int DailyWage;
    int DaysWorked;
public:
    ContractLabour(int id, string n, int days, float wages)
        : Employee(id, n) {
        DaysWorked = days;
        DailyWage = wages;
    }

    // Overriding display function
    void display() {
        float totalWages = DaysWorked * DailyWage;
        cout << "Employee Type: Contract Labour" << endl;
        cout << "Employee ID: " << EmployeeID << endl;
        cout << "Name: " << Name << endl;
        cout << "Days Worked: " << DaysWorked << endl;
        cout << "Total Wages: " << totalWages << endl<<endl;
    }
};

int main() {
    Employee* emp1 = new Salaried(101, "Alice", 22, 1500);
    Employee* emp2 = new ContractLabour(102, "Bob", 20, 800);

    cout << "Using base class pointer to derived class objects:" << endl;

    // Calling display function using base class pointer

    emp1->display(); // Will call Salaried's display function

    emp2->display(); // Will call ContractLabour's display function

    return 0;
}
```

**OUTPUT:**



```
Microsoft Windows [Version 10.0.26200.6899]
(c) Microsoft Corporation. All rights reserved.

M:\training\CPP\VIRTUAL>g++ VirtualFunction.cpp -o VirtualFunction.exe

M:\training\CPP\VIRTUAL>VirtualFunction
Using base class pointer to derived class objects:
Employee Type: Salaried Employee
Employee ID: 101
Name: Alice
Working Days: 22
Total Salary: 33000

Employee Type: Contract Labour
Employee ID: 102
Name: Bob
Days Worked: 20
Total Wages: 16000


M:\training\CPP\VIRTUAL>
```