

Write a simple program to print “Hello World” in C++.

Output:-

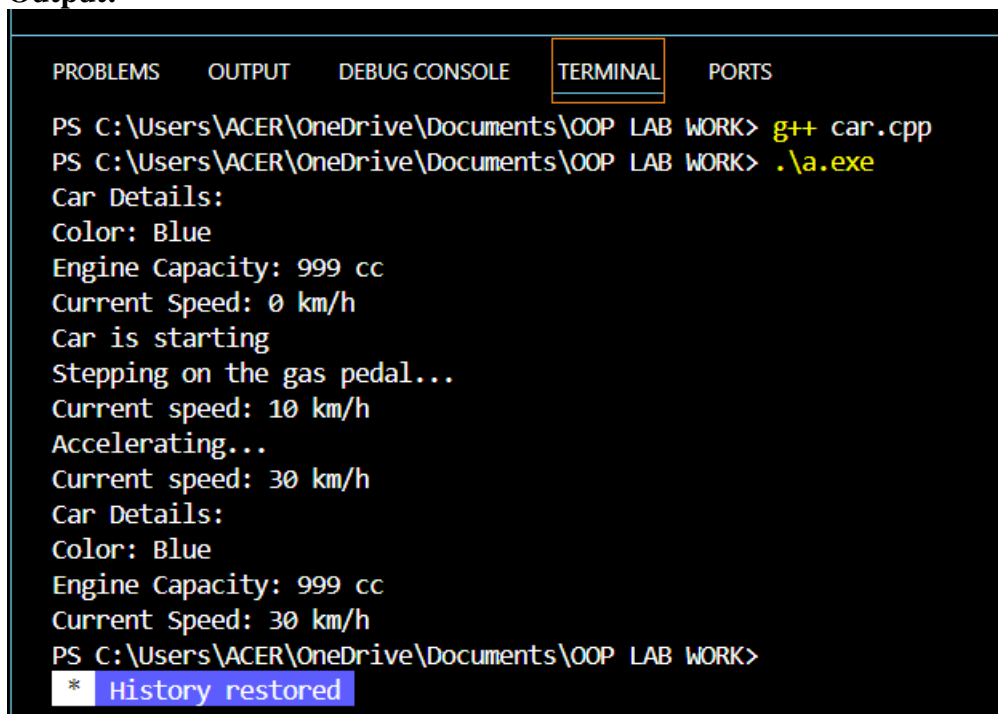
```
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ hello.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
Hello World
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> |
```

Create a class car with

a) Data member = speed, cc, color

b) Member function= start(), stop(), accelerate

Output:-



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ car.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> .\a.exe
Car Details:
Color: Blue
Engine Capacity: 999 cc
Current Speed: 0 km/h
Car is starting
Stepping on the gas pedal...
Current speed: 10 km/h
Accelerating...
Current speed: 30 km/h
Car Details:
Color: Blue
Engine Capacity: 999 cc
Current Speed: 30 km/h
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK>
* History restored
```

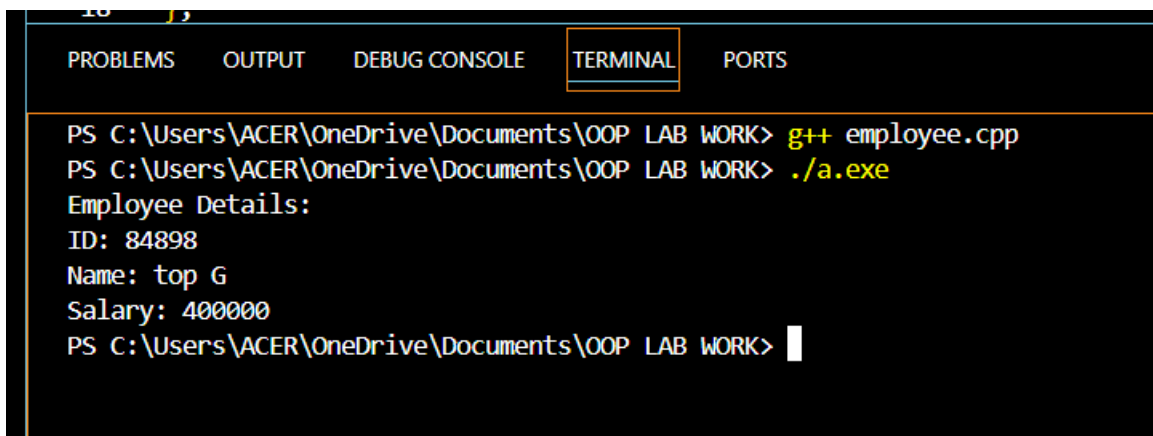
Create class called Employee with

- a) Data members = emp_id, emp_name, emp_salary**
- b) Member function= displayDetails()**

Note: Data member must be private

Initialize Data members using parameterized, constructor

Output:-



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ employee.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
Employee Details:
ID: 84898
Name: top G
Salary: 400000
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> 
```

Create a class student with following

a. Data members = student_name, student_roll, student_age

**b. Members functions = getName(), setName(), getAge(),setAge(),
getRoll(),setroll, displaydetails()**

Output:-

```
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ student.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
Name: Mandip Chaudhary
Roll Number: 44
Age: 20
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> |
```

5. Define a class bank to represent bank account with

- a. Data members = depositer_name, account_number,
account_type(as static) account_balance**
- b. Member functions = withdrawAmount(),depositAmount()**
- c. Friend function = displayDetails () to display the details.**

Output:-

```
Age: 20
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ bank.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
Depositor Name: Mandip Chaudhary
Account Number: 1234567788
Account Type: Savings
Account Balance: 900000
Deposit of 25000 successful.
Withdrawal of 20000 successful.
Depositor Name: Mandip Chaudhary
Account Number: 1234567788
Account Type: Savings
Account Balance: 905000
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> █
```

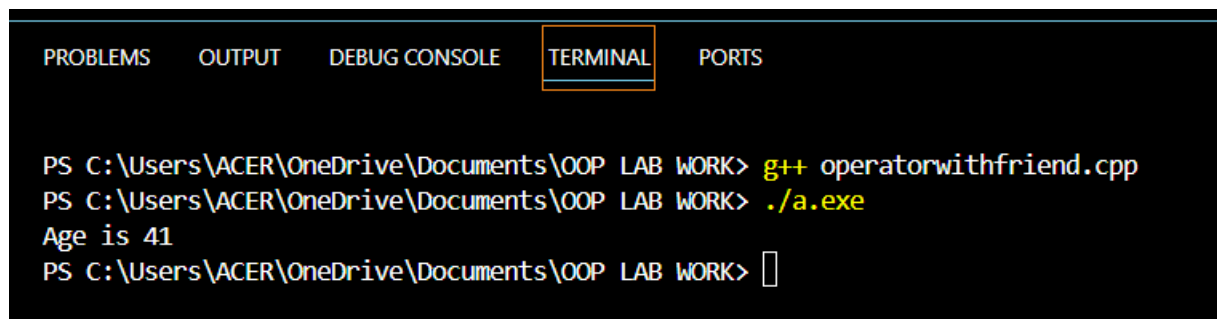

Demonstrate unary operator overloading(without friend function).

Output:-

```
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ operatorwithoutfriend.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
8
9
8
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> |
```

Demonstrate unary operation overloading (using friend function).

Output:-



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ operatorwithfriend.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
Age is 41
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> 
```

Demonstrate binary operator overloading (without friend function).

Output:-

```
Age 13 41
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ binarywithoutfriend.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
Initial rectangles:
Rectangle: Width = 8, Height = 8
Rectangle: Width = 9, Height = 9
After addition:
Rectangle: Width = 17, Height = 17
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> 
```

Demonstrate binary operator overloading (using friend function)

Output:-

```
Rectangle: Width = 17, Height = 17
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ binarywithfriend.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
Initial rectangles:
Rectangle: Width = 5, Height = 10
Rectangle: Width = 3, Height = 7
After addition:
Rectangle: Width = 8, Height = 17
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> 
```

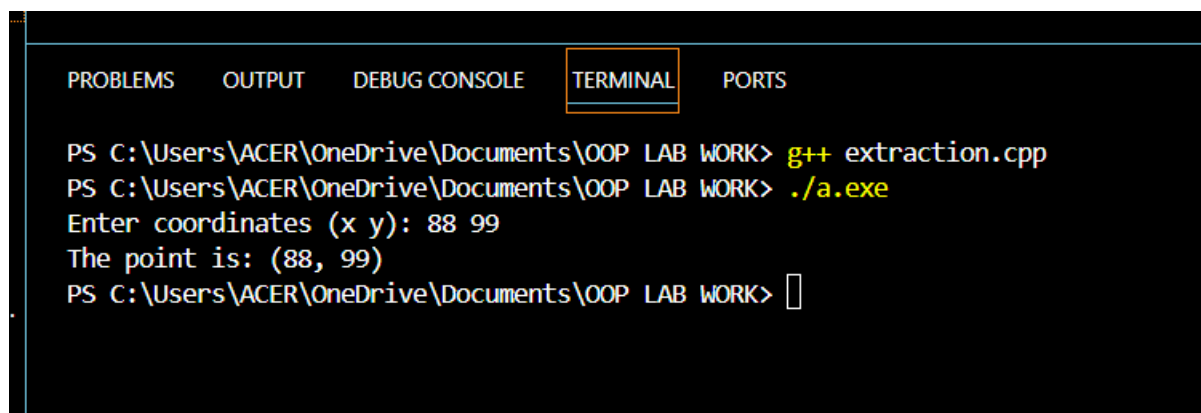
Demonstrate insertion operation overloading.

Output:-

```
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ insertion.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
(80, 90)
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> █
```

Demonstrate extraction operator overloading.

Output:-

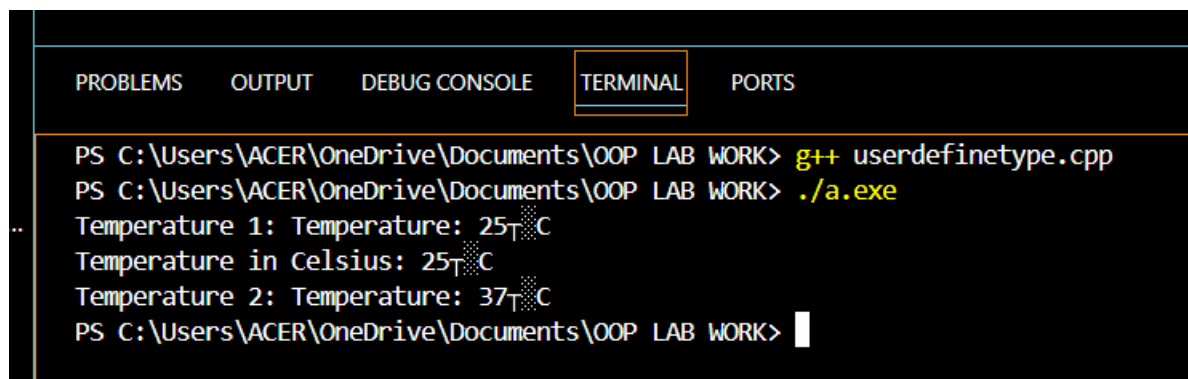


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ extraction.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
Enter coordinates (x y): 88 99
The point is: (88, 99)
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> 
```


WAP to demonstrate basic to user-defined type conversion.

Output:-



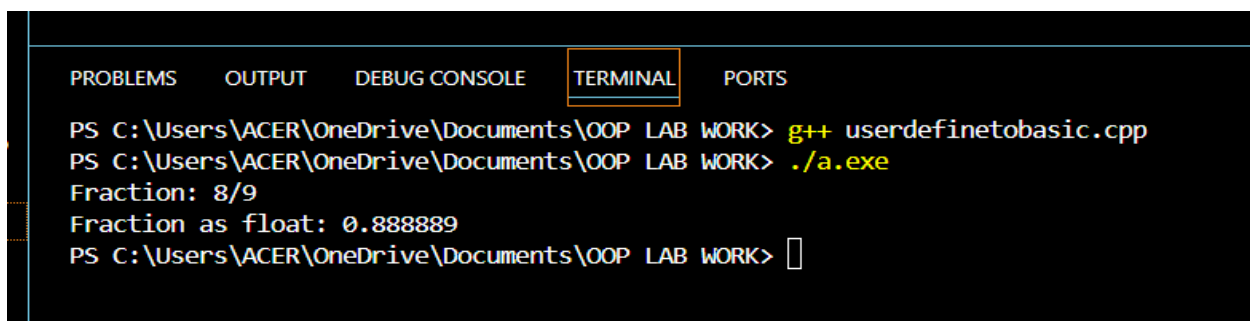
The image shows a screenshot of a C++ IDE's terminal window. The terminal has a dark background with a light blue title bar. The title bar contains five tabs: "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL" (which is selected and highlighted with a blue border), and "PORTS". The terminal content shows the following commands and output:

```
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ userdefintype.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
Temperature 1: Temperature: 25°C
Temperature in Celsius: 25°C
Temperature 2: Temperature: 37°C
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> 
```

The output of the program is displayed in the terminal, showing the temperature in Celsius for two different cases. The first case shows a temperature of 25°C, and the second case shows a temperature of 37°C. The program is compiled using g++ and executed using ./a.exe.

WAP to demonstrate user-defined to basic type conversion.

Output:-



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ userdefinetobasic.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
Fraction: 8/9
Fraction as float: 0.888889
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> 
```

WAP to demonstrate user-defined to user-defined type conversion

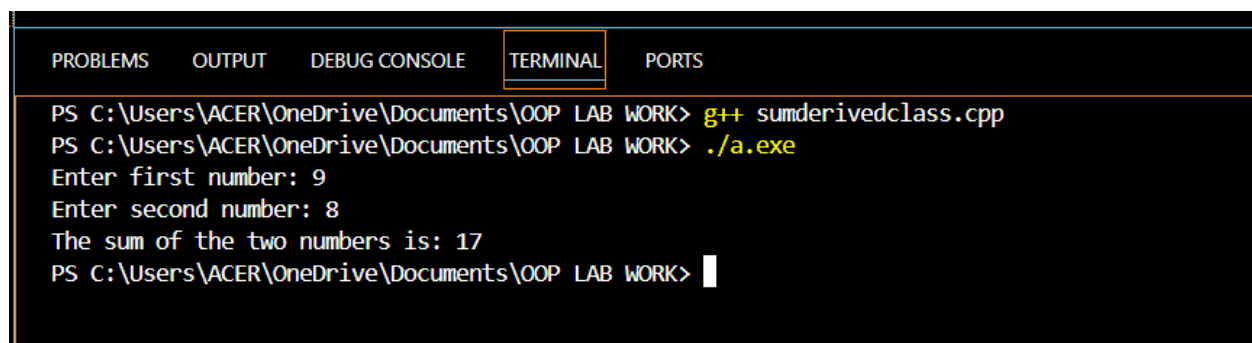
- a) Implicit conversion**
- b) Explicit conversion**

Output:-

```
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ implicitandexplicit.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
Temperature in Fahrenheit: 77°F
Temperature in Fahrenheit: 77°F
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> 
```

Write a C++ program to add two numbers using single inheritance. Accept these two numbers from the user in base class and display the sum of these two numbers in derived class.

Output:-



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ sumderivedclass.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
Enter first number: 9
Enter second number: 8
The sum of the two numbers is: 17
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> 
```

Write a C++ program to calculate the percentage of a student using multilevel inheritance. Accept the marks of three subjects in base class. A class will be derived from the above mentioned class which includes a function to find the total marks obtained and another class derived from this class which calculates and displays the percentage of student.

Output:-

```
The sum of the two numbers is: 17
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ multilevelinheritance.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
Enter marks for subject 1: 50
Enter marks for subject 2: 51
Enter marks for subject 3: 52
Total marks: 153
Percentage: 51%
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> █
```


Write a C++ program to demonstrate how a common friend function can be used to exchange the private values of two classes. (Use call by reference method).

Output:-

```
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ commonfriendexc.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
Before swapping:
Class1 value: 90
Class2 value: 80
After swapping:
Class1 value: 80
Class2 value: 90
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> 
```

Write class declarations and member function definitions for a C++ base class to represent an Employee (emp code, name). Derive two classes as Fulltime (daily rate, number of days, salary) and Parttime (number of working hours, hourly rate, salary).

Write a menu driven program to:

1. Accept the details of 'n' employees.
2. Display the details of 'n' employees.
3. Search a given Employee by emp code.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ employeefullhalf.cpp

PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe

Menu:

1. Accept details of 'n' employees
2. Display details of 'n' employees
3. Search an employee by emp-code
4. Exit

Enter your choice: 1

Enter the number of employees: 2

Enter details for Employee 1:

Employee Code: 101

Name: Mandy Chaudhary

Employee Type:

1. Fulltime
2. Parttime

Enter your choice: 1

Daily Rate: 2000

Number of Days: 30

Enter details for Employee 2:

Employee Code: 102

Name: Mandip Chaudhary

Employee Type:

1. Fulltime
2. Parttime

Enter your choice: 2

Hourly Rate: 1000

Number of Working Hours: 8

Menu:

1. Accept details of 'n' employees
2. Display details of 'n' employees
3. Search an employee by emp-code
4. Exit

Enter your choice: 2

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

Employee Details:

Employee Code: 101

Name: Mandy Chaudhary

Daily Rate: 2000

Number of Days: 30

Salary: 60000

Employee Code: 102

Name: Mandip Chaudhary

Hourly Rate: 1000

Number of Working Hours: 8

Salary: 8000

Menu:

1. Accept details of 'n' employees
2. Display details of 'n' employees
3. Search an employee by emp-code
4. Exit

Name: Mandy Chaudhary

Daily Rate: 2000

Number of Days: 30

Name: Mandy Chaudhary

Daily Rate: 2000

Name: Mandy Chaudhary

Name: Mandy Chaudhary

Name: Mandy Chaudhary

Name: Mandy Chaudhary

Daily Rate: 2000

Name: Mandy Chaudhary

Daily Rate: 2000

Name: Mandy Chaudhary

Daily Rate: 2000

Number of Days: 30

Salary: 60000

Number of Days: 30

Salary: 60000

Write a program to demonstrate ambiguity in multiple inheritance. Also show the ways to solve it using an example.

Output:-

```
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ ambiguity.cpp
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> ./a.exe
Base1's show function.
Base2's show function.
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ ambiguity1.cpp
```

Write a C++ program that demonstrates the concept of polymorphism using virtual functions. Create a base class Shape with a virtual function area(). Create two derived classes, Circle and Rectangle, each with their own implementation of the area() function. Calculate and display the area of different shapes using polymorphism.

Output:-

```
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ .\polymorphismCircleandRect.cpp ; ./a.exe
Area of Circle: 254.469
Area of Rectangle: 72
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> |
```


Create an abstract base class `Vehicle` with a pure virtual function `void start()`. Derive two classes, `Car` and `Motorcycle`, from `Vehicle`. Implement the `start()` function differently in each derived class. Write a program to create objects of both `Car` and `Motorcycle` and call their `start()` functions.

Output:-

```
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ .\virtualfuncarandbike.cpp ; .\a.exe
This is Motorcycle starting.
This is Car starting.
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> |
```

Write a program that defines an abstract base class `Animal` with a pure virtual function `void speak()`. Create two derived classes, `Dog` and `Cat`, which implement the `speak()` function. Use an array of `Animal` pointers to store instances of both `Dog` and `Cat`. Write a loop to make all animals in the array speak.

Output:-

```
This is car starting.  
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ .\virtualfunarrayanimal.cpp ; .\a.exe  
Dog is barking  
Cat is purring  
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> █
```

Create a C++ program that models a simple banking system. Implement a base class Account and derived classes SavingsAccount and CheckingAccount. Use virtual functions to perform operations like deposit, withdrawal, and interest calculation.

Output:-

```
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> g++ .\bankclassandobject.cpp ; .\a.exe
Initial Savings Account Balance: $81000.9
Deposited: $9000 | New Balance: $90000.9
Interest added: $4500.04 | New Balance: $94500.9

Initial Checking Account Balance: $81000.9
Overridden function of withdrawal from checking class
Withdrew: $2000 | New Balance: $79000.9
PS C:\Users\ACER\OneDrive\Documents\OOP LAB WORK> 
```