OOP Lab 4 : Classes, Objects, and Object Interaction in C++

Exercise 1: Create a class named **Student**.

· Declare the following data members as **private**:

– rollNo (int)

– marks (int)

· Declare the following member functions as **public**:

– setData() to assign values to the data members

– display() to print student details

In main():

· Create an object named s1

· Store and display the student details using public member functions

Ans -

Code :

```
#include<iostream>   // Include the input-output stream library
using namespace std; // Use the standard namespace
class Student{// Define a class named Student
   private :
     int rollNo;  // Private data member to store roll number
     int mark;    // Private data member to store marks

   public :
     void setData(int a,int b){
       rollNo = a;  // Assign parameter 'a' to rollNo
       mark = b;    // Assign parameter 'b' to mark
     }

     void display(){
       cout << "Roll number :" << rollNo << "\t\tMarks :" << mark;
     }
};
int main(){
    Student s1;   // Create an object 's1' of class Student
    int a, b;

    cout << "Enter Roll number :";
    cin >> a;                     // Read roll number from user

    cout << "Enter the marks :";
    cin >> b;                     // Read marks from user

    s1.setData(a, b);  // Call setData() to assign values to s1
    s1.display();      // Call display() to show student details

    return 0;
}
```

Output :

```
PS E:\25U020136_Vansh_Chadha> cd "e:\25U020136_Vansh_Chadha\" ; if ($?) { g++ lab4_q1.cpp -o lab4_q1 } ; if ($?) { .\lab4_q1 }
Enter Roll number :36
Enter the marks :94
Roll number :36         Marks :94
PS E:\25U020136_Vansh_Chadha>
```

Summary: This code defines a Student class with encapsulated data members and functions to set and display student details, then uses an object in main() to input and output a student's roll number and marks.

Exercise 2: Create a class named **Number**.

- Declare one private data member:

  – value (int)

- Declare a **default constructor** that initializes value to 10

- Declare a public member function **show()** to display the value

In main():

- Create an object named n1

- Display the initialized value

Ans -

Code :

```cpp
#include<iostream>
using namespace std;
class Number {
  private:
    int value;   // Private data member
  public:
    // Default constructor: initializes value to 10
    Number() {
        value = 10;
    }
    // Function to display the value
    void show() {
        cout << "Value is :" << value << endl;
    }
};
int main() {
    Number n1;
    n1.show();   // Displays initialized value
    return 0;
}
```

Output :

```
PS E:\25U020136_Vansh_Chadha> cd "e:\25U020136_Vansh_Chadha\" ; if ($?) { g++ lab4_q2.cpp -o lab4_q2 } ; if ($?) { .\lab4_q2 }
Value is :10
PS E:\25U020136_Vansh_Chadha>
```

Summary : This program defines a Number class with a private data member value. A default constructor initializes value to 10, and the show() function displays it. When an object is created, the constructor ensures the object starts with a meaningful default value.

Exercise 3: Create a class named **Rectangle**.

- Declare the following data members as **private**:
  - length (int)
  - width (int)
- Declare the following member functions as **public**:
  - setDimensions(int l, int w)
  - calculateArea()
  - displayArea()

In main():

- Create an object named **rect1**
- Set dimensions and display the area

Ans -

Code :

```cpp
#include<iostream>       // Include input-output stream library
using namespace std;     // Use the standard namespace
class Rectangle {
   private:
    int length;          // Private data member to store length
    int width;           // Private data member to store width
   public:
    int area;
    // Function to set dimensions of the rectangle
    void setDimension(int l, int w) {
        length = l;      // Assign parameter 'l' to length
        width = w;       // Assign parameter 'w' to width
    }
    void calculateArea() {
      area = length * width;   // Area = length x width
    }
    void displayArea() {
        cout << "Area of rectangle is :" << area;
    }
};
int main() {
    Rectangle rec1;      // Create an object 'rec1' of class Rectangle
    int a, b;            // Variables to store user input
    cout << "Enter Length :";
    cin >> a;                   // Read length from user
    cout << "Enter width :";
    cin >> b;                   // Read width from user
    rec1.setDimension(a, b);    // Set dimensions using user input
    rec1.calculateArea();       // Calculate area based on dimensions
    rec1.displayArea();         // Display the calculated area
    return 0;
}
```

Output :

```
PS E:\25U020136_Vansh_Chadha> cd "e:\25U020136_Vansh_Chadha\" ; if ($?) { g++ lab4_q3.cpp -o lab4_q3 } ; if ($?) { .\lab4_q3 }
Enter Length :23
Enter width :34
Area of rectangle is :782
PS E:\25U020136_Vansh_Chadha>
```

Summary : This program defines a Rectangle class with private members for dimensions and public functions to set values, calculate area, and display it. In main(), the user inputs dimensions, and the program outputs the rectangle's area.

Exercise 4: Create a class named **BankAccount**.

- Declare the following data member as **private**:
  - balance (double)
- Declare the following member functions as **public**:
  - setBalance(double amount)
  - getBalance()

In main():

- Create an object named **acc1**
- Set and retrieve the balance using public functions only
- Do not access the private data member directly

Ans -

Code :

```cpp
#include<iostream>        // Include input-output stream library
using namespace std;      // Use the standard namespace
// Define a class named BankAccount
class BankAccount {
   private:
    double balance;       // Private data member to store account balance
   public:
    // Function to set the balance
    void setBalance(double amount) {
        balance = amount; // Assign parameter 'amount' to balance
    }
    // Function to display the balance
    void getBalance() {
        cout << "Balance :" << balance;
    }
};
int main() {
    BankAccount acc1;     // Create an object 'acc1' of class BankAccount
    double a;             // Variable to store user input
    cout << "Enter balance :"; // Prompt user for balance
    cin >> a;                  // Read balance from user
    acc1.setBalance(a);        // Set balance using user input
    acc1.getBalance();         // Display the balance
    return 0;                  // End of program
}
```

Output :

```
PS E:\25U020136_Vansh_Chadha> cd "e:\25U020136_Vansh_Chadha\" ; if ($?) { g++ lab4_q4.cpp -o lab4_q4 } ; if ($?) { .\lab4_q4 }
Enter balance :3460.7
Balance :3460.7
PS E:\25U020136_Vansh_Chadha>
```

Summary: This program defines a BankAccount class with a private balance and public functions to set and display it. In main(), the user inputs a balance, which is stored in the object and then displayed. It demonstrates encapsulation and basic class usage in C++.

Exercise 5: Create a class named **Employee**.

- Declare the following data members as **private**:
  - empId (int)
  - salary (float)
- Declare the following member functions as **public**:
  - setData(int id, float sal)
  - showData()

In main():

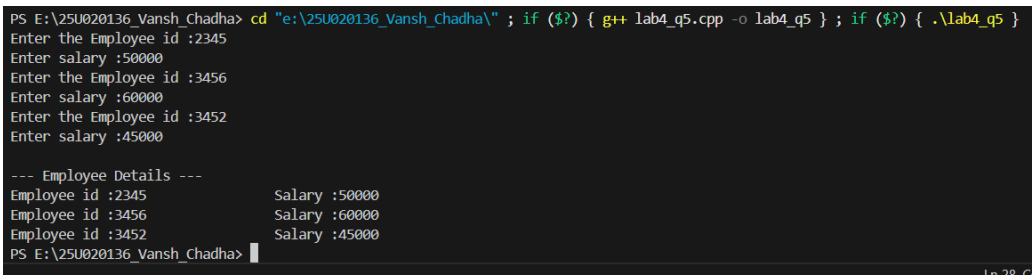
- Create an array of three objects named **emp[3]**
- Store and display employee details using loops

Ans -

Code:-

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

class Employee {
private:
    int empId;    // [cite: 51]
    float salary; // [cite: 52]

public:
    // Set employee data [cite: 56]
    void setData(int id, float sal) {
        empId = id;
        salary = sal;
    }

    // Show employee data [cite: 57]
    void showData() {
        cout << "ID: " << empId << " | Salary: " << salary << endl;
    }
};

int main() {
    Employee emp[3]; // Create array of 3 objects [cite: 58]
    int id;
    float sal;

    // Loop to input details [cite: 59]
    cout << "--- Enter details for 3 Employees ---" << endl;
    for(int i = 0; i < 3; i++) {
        cout << "Enter ID and Salary for Employee " << (i + 1) << ": ";
        cin >> id >> sal;
        emp[i].setData(id, sal);
    }

    // Loop to display details [cite: 59]
    cout << "\n--- Employee Records ---" << endl;
    for(int i = 0; i < 3; i++) {
        emp[i].showData();
    }

    return 0;
}
```

Output :



Summary: This program defines an Employee class with private data members for ID and salary. It uses an array of objects to store details of multiple employees, takes input from the user, and displays all employee details. It demonstrates encapsulation, arrays of objects, and class methods in C++.

Exercise 6: Create two classes named **Author and Book**.

Class Author

· Private data member:

– name (string)

· Public member functions:

– setName(string n)

– getName()

Class Book

· Private data member:

– title (string)

· Public member function:

– display(Author a) to display book title and author name

In main():

· Create objects a1 (Author) and b1 (Book)

· Pass the Author object to the Book object

Ans -

Code :

```cpp
#include<iostream>
using namespace std;
class Author {
   private:
      string name;   // to store Author name
   public:
      void setName(string n) {
        name = n;
      }
      string getName() {   // Return name instead of printing
        return name;
      }
};
class Book {
   private:
     string title;   // Book title
  public:
     void setTitle(string t) {
        title = t;
     }
     void display(Author a) {
        cout << "\nAuthor : " << a.getName() << " \tTitle : " << title;
     }
};
int main() {
    Author a1;
    Book b1;
    string a, b;
    cout << "Enter the Author : ";
    cin >> a;
    cout << "Enter the Title : ";
    cin >> b;
    a1.setName(a);      // Set author name
    b1.setTitle(b);     // Set book title
    b1.display(a1);     // Display both together
    return 0;
}
```

```
PS E:\25U020136_Vansh_Chadha> cd "e:\25U020136_Vansh_Chadha\" ; if ($?) { g++ lab4_q6.cpp -o lab4_q6 } ; if ($?) { .\lab4_q6 }
Enter the Author : William
Enter the Title : Hamlet

Author : William        Title : Hamlet
PS E:\25U020136_Vansh_Chadha>
```

Summary: This program defines two classes, Author and Book, with private data members and public functions. The Book class interacts with the Author class to display both the author's name and the book's title, showing how objects can collaborate in C++.

Exercise 7: Create a class named **Demo**.

· Declare a private data member:

– num (int)

· Declare public member functions:

– set(int num) using the this pointer

– show() to display the value

In main():

· Create an object named **d1**

· Call both functions

Ans -

Code :

```
#include<iostream>
using namespace std;
class Demo {
    private:
        int num;   // Private data member
    public:
        // Function to set the value of num using 'this' pointer
        void set(int num) {
            this->num = num;   // 'this' points to the current object
        }
        // Function to display the value of num
        void show() {
            cout << "The number is: " << num << endl;
        }
};
int main() {
    Demo d1;   // Create object of Demo
    int a;
    cout << "Enter a number : ";
    cin >> a;          // Take input from user
    d1.set(a);         // Store input in object using 'this' pointer
    d1.show();         // Display stored value
    return 0;
}
```

Output :

```
PS E:\25U020136_Vansh_Chadha> cd "e:\25U020136_Vansh_Chadha\" ; if ($?) { g++ lab4_q7.cpp -o lab4_q7 } ; if ($?) { .\lab4_q7 }
Enter a number : 34
The number is: 34
PS E:\25U020136_Vansh_Chadha>
```

Summary: This program defines a Demo class with a private integer and uses the this pointer in set() to correctly assign values when parameter names shadow member variables. It shows how encapsulation and controlled access work in C++.

Exercise 8: Create a class named **Sample**.

- Declare a public data member:
  - x (int)

Write a function named **modify()**:

- Accepts a **Sample** object by value
- Modifies the value of **x**

In main():

- Create an object named **obj1**
- Show that changes inside the function do not affect the original object
- Explain why the object address and its copy are different

Ans -

Code :

```cpp
#include<iostream>
using namespace std;
class Sample {
   public:
      int x;   // Public data member
};
// Function that accepts Sample object by value , not part of class Sample
void modify(Sample s) {
    cout << "\n[Inside modify()] Original x = " << s.x;
    s.x = 100;   // Modify the copy
    cout << "\n[In modify()] Modified x = " << s.x;
    cout << "\nAddress of object in modify(): " << &s << endl;
}
int main() {
    Sample obj1;     // Create object
    obj1.x = 10;     // Initialize x
    cout << "Before modify(), obj1.x = " << obj1.x;
    cout << "\nAddress of obj1 in main(): " << &obj1 << endl;
    modify(obj1);    // Pass object by value
    cout << "\nAfter modify(), obj1.x = " << obj1.x << endl;
    return 0;
}
```

Output :

```
PS E:\25U020136_Vansh_Chadha> cd "e:\25U020136_Vansh_Chadha\" ; if ($?) { g++ lab4_q8.cpp -o lab4_q8 } ; if ($?) { .\lab4_q8 }
Before modify(), obj1.x = 10
Address of obj1 in main(): 0x61ff0c

[Inside modify()] Original x = 10
[In modify()] Modified x = 100
Address of object in modify(): 0x61fef0

After modify(), obj1.x = 10
PS E:\25U020136_Vansh_Chadha>
```

Summary: This program shows that when an object is passed by value, modifications inside the function affect only the copy, not the original object. The printed addresses highlight the difference between the original object and its copy.

Exercise 9: Using the same Sample class from Exercise 8:

- Write a function named **modifyByAddress()**
- Accept a pointer to a **Sample** object
- Modify the data member using pointer notation

In main():

- Create an object named **obj2**
- Pass its address to the function
- Show that the changes affect the original object
- Access object data using both object and pointer

Ans -

Code :

Output :

```cpp
#include<iostream>
using namespace std;
class Sample {
   public:
      int x;   // Public data member
};
// Function that accepts a pointer to a Sample object
void modifyByAddress(Sample *s) {
    cout << "\n[In modifyByAddress()] Original x = " << s->x;
    s->x = 200;   // Modify using pointer notation
    cout << "\n[Ins modifyByAddress()] Modified x = " << s->x;
    cout << "\nAddress of object in modifyByAddress(): " << s << endl;
}
int main() {
    Sample obj2;     // Create object
    obj2.x = 20;     // Initialize x
    cout << "Before modifyByAddress(), obj2.x = " << obj2.x;
    cout << "\nAddress of obj2 in main(): " << &obj2 << endl;
    // Pass address of obj2 to the function
    modifyByAddress(&obj2);
    // Show that changes affect the original object
    cout << "\nAfter modifyByAddress(), obj2.x = " << obj2.x << endl;
    // Access object data using both object and pointer
    Sample *ptr = &obj2;   // Pointer to obj2
    cout << "\nAccess via object: obj2.x = " << obj2.x;
    cout << "\nAccess via pointer: ptr->x = " << ptr->x << endl;
    return 0;
}
```

Output :

```
PS E:\25U020136_Vansh_Chadha> cd "e:\25U020136_Vansh_Chadha\" ; if ($?) { g++ lab4_q9.cpp -o lab4_q9 } ; if ($?) { .\lab4_q9 }
Before modifyByAddress(), obj2.x = 20
Address of obj2 in main(): 0x61ff08

[In modifyByAddress()] Original x = 20
[Ins modifyByAddress()] Modified x = 200
Address of object in modifyByAddress(): 0x61ff08

After modifyByAddress(), obj2.x = 200

Access via object: obj2.x = 200
Access via pointer: ptr->x = 200
PS E:\25U020136_Vansh_Chadha>
```

Summary: This program shows how passing an object by address allows functions to modify the original object. It demonstrates pointer notation (s->x) and how both object and pointer access lead to the same data.