1. **Create a class Employee with data members employeeID, name, and salary. Add member functions to read and display these details. Write a program to input and display any number of employees as desired by the user.**

**Answer:**

#include <iostream>

using namespace std;

class Employee {

public:

int employeeID;

string name;

double salary;

void readDetails() {

cout << "Enter Employee ID: ";

cin >> employeeID;

cout << "Enter Name: ";

cin >> name;

cout << "Enter Salary: ";

cin >> salary;

}

void displayDetails() {

cout << "Employee ID: " << employeeID << endl;

cout << "Name: " << name << endl;

cout << "Salary: " << salary << endl;

}

};

int main() {

int numE;

cout << "Enter the number of employees: ";

cin >> numE;

Employee employees[numE];

for (int i = 0; i < numE; i++) {

cout << "Enter details for Employee " << i + 1 << endl;

employees[i].readDetails();

}

cout << "Details of all employees:" << endl;

for (int i = 0; i < numE; i++) {

cout << "Employee " << i + 1 << " details:" << endl;

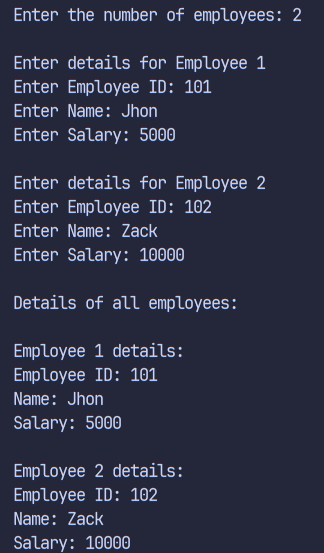
employees[i].displayDetails();

}

return 0;

}

**Output:**



1. **Design a class Student with members name, roll, and marks. Provide appropriate methods to read and display data.**

**Answer:**

#include <iostream>

Using namespace std;

class Student {

public:

string name;

int roll;

float marks;

void readData() {

cout << "Enter name: ";

cin >> name;

cout << "Enter roll number: ";

cin >> roll;

cout << "Enter marks: ";

cin >> marks;

}

void displayData() {

cout << "Name: " << name << endl;

cout << "Roll Number: " << roll << endl;

cout << "Marks: " << marks << endl;

}

};

int main() {

Student student;

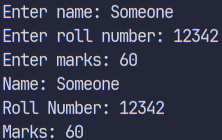
student.readData();

student.displayData();

return 0;

}

**Output:**



1. **Write a program designing a class Customer with member variables firstName,lastName, address, and phoneNo. Add member functions to accept input and display these variables.**

**Answer:**

#include <iostream>

using namespace std;

class Customer {

public:

string firstName;

string lastName;

string address;

string phoneNo;

void acceptInput() {

cout << "Enter first name: ";

cin >> firstName;

cout << "Enter last name: ";

cin >> lastName;

cout << "Enter address: ";

cin >> address;

cout << "Enter phone number: ";

cin >> phoneNo;

}

void displayInfo() {

cout << "First Name: " << firstName << endl;

cout << "Last Name: " << lastName << endl;

cout << "Address: " << address << endl;

cout << "Phone Number: " << phoneNo << endl;

}

};

int main() {

Customer c;

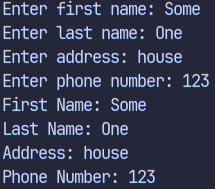
c.acceptInput();

c.displayInfo();

return 0;

}

**Output:**



1. **Create a class Information to store name and address of students. Store information for two students and write a function to swap the contents of these two objects.**

**Answer:**

#include<iostream>

using namespace std;

class Info{

public:

string name;

string address;

};

void swapInfo(Info &info1, Info &info2) {

Info temp = info1;

info1 = info2;

info2 = temp;

cout << "Swapped information successfully." << endl;

}

int main() {

Info student1, student2;

cout << "Enter name for student 1 and address: ";

cin >> student1.name >> student1.address;

cout << "Enter name for student 2 and address: ";

cin >> student2.name >> student2.address;

swapInfo(student1, student2);

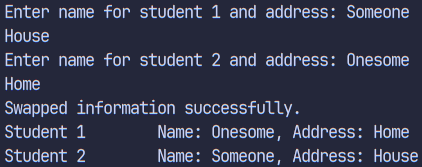
cout << "Student 1 \t Name: " << student1.name << ", Address: " << student1.address << endl;

cout << "Student 2 \t Name: " << student2.name << ", Address: " << student2.address << endl;

return 0;

}

**Output:**



1. **Design a class to represent a bank account with members: depositor’s name, account number, and account type. Include member functions to initialize and display the depositor’s name and account type.**

**Answer:**

#include <iostream>

using namespace std;

class BankAccount {

public:

string depositorName;

int accountNumber;

string accountType;

void initialize(string name, int number, string type) {

depositorName = name;

accountNumber = number;

accountType = type;

}

void displayInfo() {

cout << "Depositor's Name: " << depositorName << endl;

cout << "Account Number: " << accountNumber << endl;

cout << "Account Type: " << accountType << endl;

}

};

int main() {

BankAccount account;

string name, type;

int number;

cout << "Enter depositor's name, account number, and account type: ";

cin >> name >> number >> type;

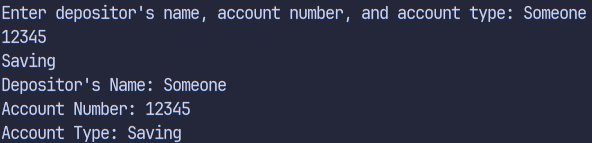
account.initialize(name, number, type);

account.displayInfo();

return 0;

}

**Output:**



1. **Write a program to design a class Rectangle with constructors and member functions to calculate area and perimeter.**

**Answer:**

#include <iostream>

using namespace std;

class Rectangle {

private:

float length;

float width;

public:

Rectangle(float l, float w) {

length = l;

width = w;

}

float calcA() {

return length \* width;

}

float calcP() {

return 2 \* (length + width);

}

};

int main() {

float l, w;

cout << "Enter length and width of the rectangle: ";

cin >> l >> w;

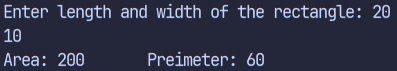
Rectangle R(l, w);

cout<<"Area: "<<R.calcA()<<"\tPreimeter: " <<R.calcP()<<endl;

return 0;

}

**Output:**



1. **Write a program demonstrating that destructors execute in the reverse order of constructors. Display the corresponding object IDs during construction and destruction.**

**Answer:**

#include <iostream>

using namespace std;

class exmp {

private:

int id;

public:

exmp(int i) {

id = i;

cout << "Constructing object with ID: " << id << endl;

}

~exmp() {

cout << "Destructing object with ID: " << id << endl;

}

};

int main() {

exmp obj1(1);

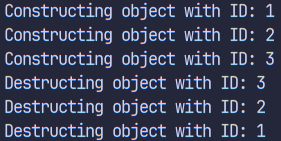
exmp obj2(2);

exmp obj3(3);

return 0;

}

**Output:**



1. **Create a class Time with members for hours, minutes, and seconds. Write a member function AddTime() that adds two Time objects passed as arguments and returns the result.**

**Answer:**

#include <iostream>

using namespace std;

class Time {

public:

int h;

int m;

int s;

Time AddTime(Time t1, Time t2) {

Time r;

r.h = t1.h + t2.h;

r.m = t1.m + t2.m;

r.s = t1.s + t2.s;

if (r.s >= 60) {

r.m += r.s / 60;

r.s %= 60;

}

if (r.m >= 60) {

r.h += r.m / 60;

r.m %= 60;

}

return r;

}

};

int main() {

Time t1, t2;

cout << "Enter hours, minutes, and seconds for t1: ";

cin >> t1.h >> t1.m >> t1.s;

cout << "Enter hours, minutes, and seconds for t2: ";

cin >> t2.h >> t2.m >> t2.s;

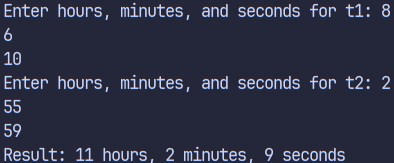
Time r = t1.AddTime(t1, t2);

cout << "Result: " << r.h << " hours, " << r.m << " minutes, " << r.s << " seconds" << endl;

return 0;

}

**Output:**



1. **Design a class LandMeasure that stores Ropani, Ana, Paisa, and Dam. Write a member function to add two LandMeasure objects and return their sum as a new object. Nepali Land Measurement Reference: 1 Ropani = 16 Ana 1 Ana = 4 Paisa 1 Paisa = 4 Dam**

**Answer:**

#include<iostream>

using namespace std;

class LM {

public:

int Ropani;

int Ana;

int Paisa;

int Dam;

LM(int r, int a, int p, int d) {

Ropani = r;

Ana = a;

Paisa = p;

Dam = d;

}

LM calculateSum(LM lm) {

int total = Ropani \* 16 \* 4 \* 4 + Ana \* 4 \* 4 + Paisa \* 4 + Dam;

int totalLM = lm.Ropani \* 16 \* 4 \* 4 + lm.Ana \* 4 \* 4 + lm.Paisa \* 4 + lm.Dam;

int sum = total + totalLM;

int newRopani = sum / (16 \* 4 \* 4);

sum %= (16 \* 4 \* 4);

int newAna = sum / (4 \* 4);

sum %= (4 \* 4);

int newPaisa = sum / 4;

int newDam = sum % 4;

return LM(newRopani, newAna, newPaisa, newDam);

}

};

int main() {

int r1, a1, p1, d1, r2, a2, p2, d2;

cout << "Enter Ropani for LM 1, Ana for LM 1, Paisa for LM 1, Dam for LM 1: ";

cin >> r1 >> a1 >> p1 >> d1;

cout << "Enter Ropani for LM 2, Ana for LM 2, Paisa for LM 2, Dam for LM 2: ";

cin >> r2 >> a2 >> p2 >> d2;

LM inputLM1(r1, a1, p1, d1);

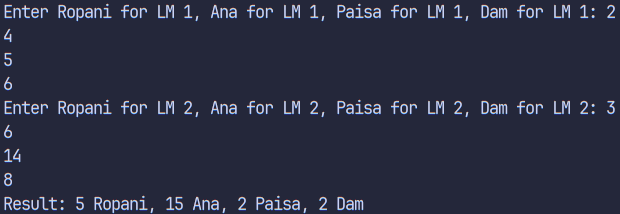
LM inputLM2(r2, a2, p2, d2);

LM CalcR = inputLM1.calculateSum(inputLM2);

cout << "Result: " << CalcR.Ropani << " Ropani, " << CalcR.Ana << " Ana, " << CalcR.Paisa << " Paisa, " << CalcR.Dam << " Dam" << endl;

return 0;

} **Output:**



1. **Create a class Employee and add a member function to increase the salary of each employee by 10%. Read and display the details of all employees before and after the salary increase.**

**Answer:**

#include <iostream>

using namespace std;

class Employee {

public:

string name;

int id;

float salary;

void increaseSalary() {

salary \*= (float)1.1;

}

void readData() {

cout << "Enter name: ";

cin >> name;

cout << "Enter ID: ";

cin >> id;

cout << "Enter salary: ";

cin >> salary;

}

void displayInfo() {

cout << "Name: " << name << endl;

cout << "ID: " << id << endl;

cout << "Salary: " << salary << endl;

}

};

int main() {

const int MAX\_EMPLOYEES = 100;

Employee employees[MAX\_EMPLOYEES];

int j;

cout << "Enter the number of employees (max 100): ";

cin >> j;

if (j > MAX\_EMPLOYEES) {

cout << "Number exceeds maximum limit. Setting to " << MAX\_EMPLOYEES << endl;

j = MAX\_EMPLOYEES;

}

for (int i = 0; i < j; i++) {

employees[i].readData();

}

cout << "Details of all employees before salary increase:" << endl;

for (int i = 0; i < j; i++) {

employees[i].displayInfo();

}

for (int i = 0; i < j; i++) {

employees[i].increaseSalary();

}

cout << "Details of all employees after salary increase:" << endl;

for (int i = 0; i < j; i++) {

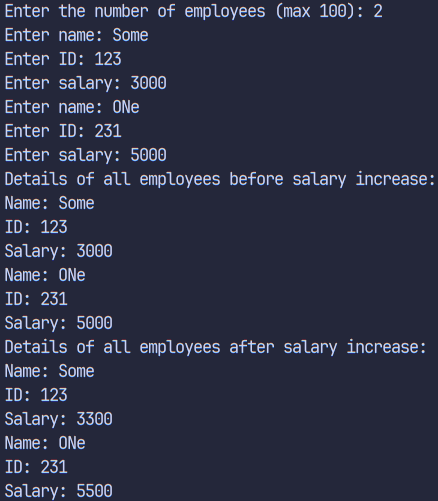
employees[i].displayInfo();

}

return 0;

}

**Output:**



1. **Write a program to find the area of a square and a rectangle using classes. Initialize the objects dynamically using pointers.**

**Answer:**

#include <iostream>

using namespace std;

class Square {

private:

float l;

public:

Square(float length) {

l = length;

}

float calculateArea() {

return l \* l;

}

};

class Rectangle {

private:

float l;

float b;

public:

Rectangle(float length, float width) {

l = length;

b = width;

}

float calculateArea() {

return l \* b;

}

};

int main() {

float sideLength;

cout << "Enter side length for square: ";

cin >> sideLength;

Square\* square = new Square(sideLength);

float length, width;

cout << "Enter length and width for rectangle: ";

cin >> length >> width;

Rectangle\* rectangle = new Rectangle(length, width);

cout << "Area of square: " << square->calculateArea() << endl;

cout << "Area of rectangle: " << rectangle->calculateArea() << endl;

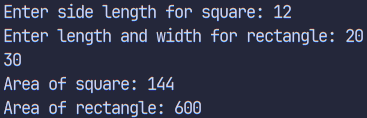
delete square;

delete rectangle;

return 0;

}

**Output:**



1. **Write a program defining an inline member function calculateVolume() outside the class using the inline keyword for a class Box.**

**Answer:**

#include <iostream>

using namespace std;

class Box {

private:

float len;

float bre;

float hei;

public:

Box(float l, float w, float h) {

len = l;

bre = w;

hei = h;

}

inline float calculateVolume(float l, float w, float h) {

return l\* w\* h;

}

};

int main() {

float l, w, h;

cout << "Enter length, width, and height of the box: ";

cin >> l >> w >> h;

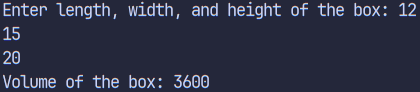
Box box(l, w, h);

cout << "Volume of the box: " << box.calculateVolume(l, w, h) << endl;

return 0;

}

**Output:**



### **Discussion**

The lab covered essential OOP concepts in C++ including classes, objects, constructors, destructors, dynamic memory allocation, inline functions, and reference handling. Through practical examples, we implemented real-world scenarios like employee management, banking systems, and geometrical calculations. These programs demonstrated how data abstraction and encapsulation enhance code structure and reusability.

### **Conclusion**

This lab helped in understanding the core principles of object-oriented programming in C++. We learned how to use classes and member functions to manage data efficiently and how concepts like constructors, destructors, and dynamic memory improve program design and performance.