# LAB QUESTIONS

Q1. 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.

#include <iostream>

#include <string>

using namespace std;

class Employee {

int employeeID;

string name;

float salary;

public:

void readData() {

cout << "Enter Employee ID: ";

cin >> employeeID;

cin.ignore();

cout << "Enter Name: ";

getline(cin, name);

cout << "Enter Salary: ";

cin >> salary;

}void displayData() {

cout << "Employee ID: " << employeeID << ", Name: " << name << ", Salary: Rs. " << salary << endl;

}

};

int main() {

int n;

cout << "number of empoloyees ";

cin >> n;

Employee\* e = new Employee[n];

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

cout << "\nEnter details for employee " << (i + 1) << ":\n";

e[i].readData(); }

cout << "\nEmployee Details:\n";

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

e[i].displayData();

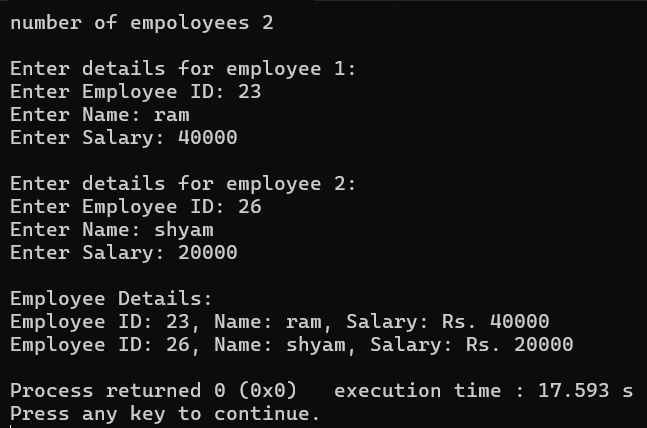
}

delete[] e;

return 0;

}

Output:



Q2. Design a class Student with members name, roll, and marks. Provide appropriate methods to read and display data.

#include <iostream>

#include <string>

using namespace std;

class Student {

string name;

int roll;

float marks;

public:

void readData() {

cout << "Enter student name: ";

getline(cin, name);

cout << "Enter roll number: ";

cin >> roll;

cout << "Enter marks: ";

cin >> marks;

cin.ignore();

}

void displayData() {

cout << "Name: " << name << ", Roll No: " << roll << ", Marks: " << marks << endl;

}

};

int main() {

Student s;

s.readData();

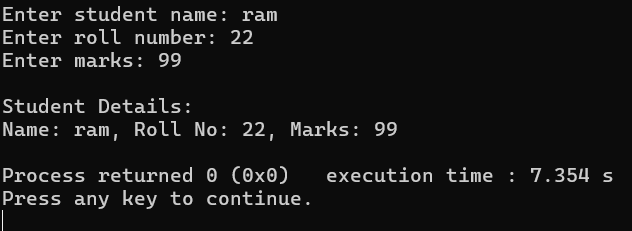
cout << "\nStudent Details:\n";

s.displayData();

return 0;

}

Output:



Q3. 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.

#include <iostream>

#include <string>

using namespace std;

class Customer {

string firstName;

string lastName;

string address;

string phoneNo;

public:

void readData() {

cout << "Enter First Name: ";

getline(cin, firstName);

cout << "Enter Last Name: ";

getline(cin, lastName);

cout << "Enter Address: ";

getline(cin, address);

cout << "Enter Phone No: ";

getline(cin, phoneNo);

}

void displayData() {

cout << "\nCustomer Details:\n";

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

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

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

cout << "Phone No: " << phoneNo << endl;

}

};

int main() {

Customer c;

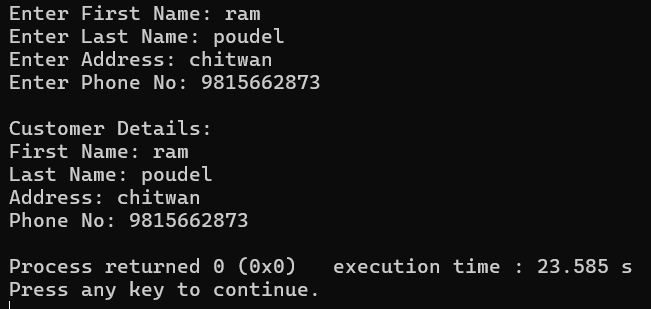
c.readData();

c.displayData();

return 0;

}

Output:



Q4. 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.

#include <iostream>

#include <string>

using namespace std;

class Information {

string name;

string address;

public:

void readData() {

cout << "Enter name: ";

getline(cin, name);

cout << "Enter address: ";

getline(cin, address);

}

void displayData() {

cout << "Name: " << name << ", Address: " << address << endl;

}

void swapInfo(Information &obj) {

string tempName = name;

name = obj.name;

obj.name = tempName;

string tempAddress = address;

address = obj.address;

obj.address = tempAddress;

}

};

int main() {

Information s1, s2;

cout << "Enter details of student 1:\n";

s1.readData();

cout << "\nEnter details of student 2:\n";

s2.readData();

cout << "\nBefore swapping:\n";

cout << "Student 1: ";

s1.displayData();

cout << "Student 2: ";

s2.displayData();

s1.swapInfo(s2);

cout << "\nAfter swapping:\n";

cout << "Student 1: ";

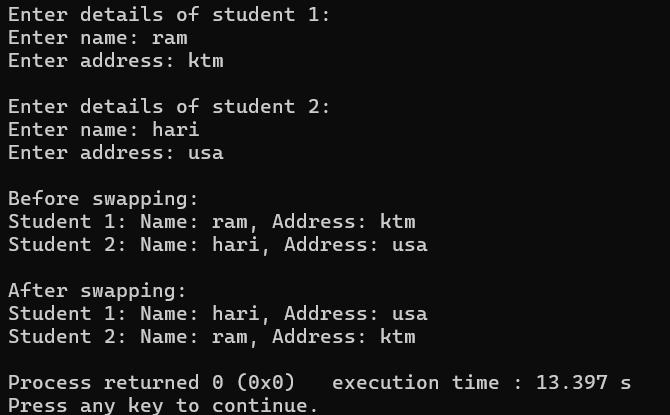
s1.displayData();

cout << "Student 2: ";

s2.displayData();

return 0;

}



Q5. Write a program to design a class Rectangle with constructors and member functions to calculate area and perimeter.

#include <iostream>

using namespace std;

class Rectangle {

double length;

double width;

public:

Rectangle() {

length = 0;

width = 0;

}

Rectangle(double l, double w) {

length = l;

width = w;

}

double area() {

return length \* width;

}

double perimeter() {

return 2 \* (length + width);

}

void display() {

cout << "Length: " << length << ", Width: " << width << endl;

cout << "Area: " << area() << endl;

cout << "Perimeter: " << perimeter() << endl;

}

};

int main() {

double l, w;

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

cin >> l >> w;

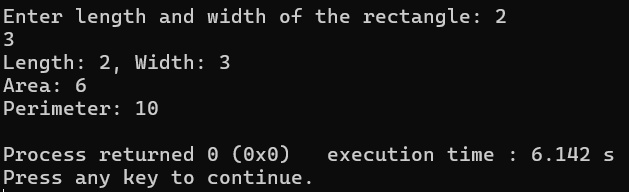
Rectangle rect(l, w);

rect.display();

return(0);

}

Output:



Q6. Write a program demonstrating that destructors execute in the reverse order of constructors. Display the corresponding object IDs during construction and destruction.

#include <iostream>

using namespace std;

class Demo {

int id;

public:

Demo(int i) {

id = i;

cout << "Constructor called for object " << id << endl;

}

~Demo() {

cout << "Destructor called for object " << id << endl;

}

};

int main() {

cout << "Creating objects...\n";

Demo d1(1);

Demo d2(2);

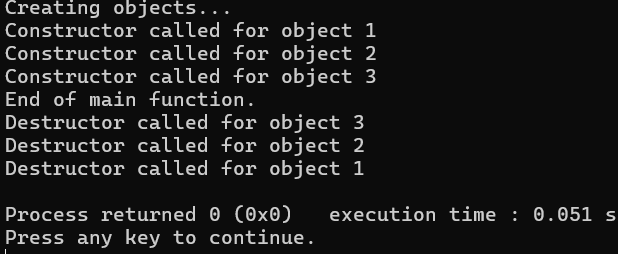
Demo d3(3);

cout << "End of main function.\n";

return 0;

}

Output.



Q7. 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.

#include <iostream>

using namespace std;

class Time {

int hours;

int minutes;

int seconds;

public:

// Member function to read time

void readTime() {

cout << "Enter hours: ";

cin >> hours;

cout << "Enter minutes: ";

cin >> minutes;

cout << "Enter seconds: ";

cin >> seconds;

}

void displayTime() {

cout << hours << "h " << minutes << "m " << seconds << "s" << endl;

}

Time AddTime(Time t2) {

Time temp;

temp.seconds = seconds + t2.seconds;

temp.minutes = minutes + t2.minutes + (temp.seconds / 60);

temp.seconds %= 60;

temp.hours = hours + t2.hours + (temp.minutes / 60);

temp.minutes %= 60;

return temp;

}

};

int main() {

Time t1, t2, sum;

cout << "Enter first time:\n";

t1.readTime();

cout << "\nEnter second time:\n";

t2.readTime();

sum = t1.AddTime(t2);

cout << "\nFirst Time: ";

t1.displayTime();

cout << "Second Time: ";

t2.displayTime();

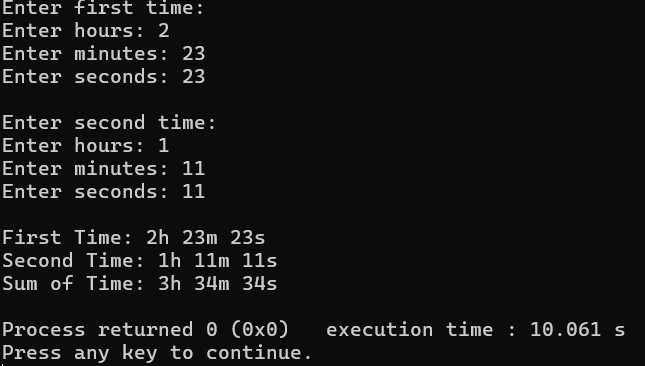
cout << "Sum of Time: ";

sum.displayTime();

return 0;

}

Output.



Q8. 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. Refrence.

1 Ropani = 16 Ana

1 Ana = 4 Paisa

1 Paisa = 4 Dam

#include <iostream>

using namespace std;

class LandMeasure {

int ropani;

int ana;

int paisa;

int dam;

public:

void read() {

cout << "Enter Ropani: ";

cin >> ropani;

cout << "Enter Ana: ";

cin >> ana;

cout << "Enter Paisa: ";

cin >> paisa;

cout << "Enter Dam: ";

cin >> dam;

}

void display() {

cout << ropani << " Ropani, " << ana << " Ana, "

<< paisa << " Paisa, " << dam << " Dam" << endl;

}

LandMeasure add(LandMeasure l2) {

LandMeasure result;

result.dam = dam + l2.dam;

result.paisa = paisa + l2.paisa + result.dam / 4;

result.dam %= 4;

result.ana = ana + l2.ana + result.paisa / 4;

result.paisa %= 4;

result.ropani = ropani + l2.ropani + result.ana / 16;

result.ana %= 16;

return result;

}

};

int main() {

LandMeasure l1, l2, sum;

cout << "Enter first land measurement:\n";

l1.read();

cout << "\nEnter second land measurement:\n";

l2.read();

sum = l1.add(l2);

cout << "\nFirst Land: ";

l1.display();

cout << "Second Land: ";

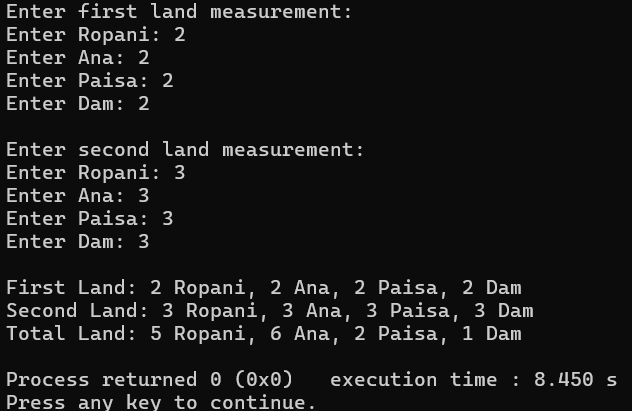
l2.display();

cout << "Total Land: ";

sum.display();

return 0;

}



Q9. 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.

#include <iostream>

using namespace std;

class Employee {

int employeeID;

string name;

float salary;

public:

void read() {

cout << "Enter Employee ID: ";

cin >> employeeID;

cout << "Enter Name: ";

cin.ignore(); getline(cin, name);

cout << "Enter Salary: ";

cin >> salary;

}

void display() {

cout << "ID: " << employeeID << ", Name: " << name << ", Salary: Rs. " << salary << endl;

}

void increaseSalary() {

salary += salary \* 0.10;

}

};

int main() {

int n;

cout << "Enter number of employees: ";

cin >> n;

Employee emp[n];

cout << "\nEnter details of employees:\n";

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

cout << "\nEmployee " << i + 1 << ":\n";

emp[i].read();

}

cout << "\nDetails before increase salary:\n";

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

emp[i].display();

}

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

emp[i].increaseSalary();

}

cout << "\nDetails after 10% increase salary:\n";

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

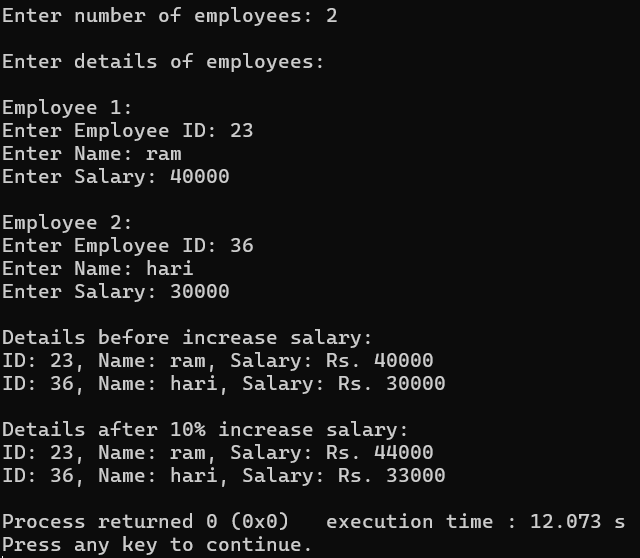
emp[i].display();

}

return 0;

}

Output.



Q10. 10.Write a program to find the area of a square and a rectangle using classes. Initialize the objects dynamically using pointers.

#include <iostream>

using namespace std;

class Square {

float side;

public:

void read() {

cout << "Enter the side of the square: ";

cin >> side;

}

float area() {

return side \* side;

}

};

class Rectangle {

float length, breadth;

public:

void read() {

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

cin >> length >> breadth;

}

float area() {

return length \* breadth;

}

};

int main() {

Square\* sq = new Square;

Rectangle\* rect = new Rectangle;

cout << "Square:\n";

sq->read();

cout << "\nRectangle:\n";

rect->read();

cout << "\nArea of Square = " << sq->area() << endl;

cout << "Area of Rectangle = " << rect->area() << endl;

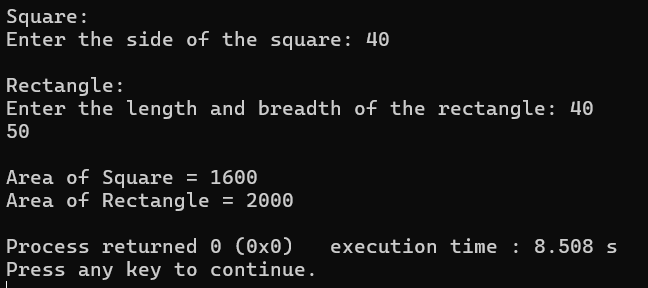
delete sq;

delete rect;

return 0;

}

Output.



Q11.Write a program defining an inline member function calculateVolume() outside the class using the inline keyword for a class Box.

#include <iostream>

using namespace std;

class Box {

float length, breadth, height;

public:

void read() {

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

cin >> length >> breadth >> height;

}

inline float calculateVolume();

};

inline float Box::calculateVolume() {

return length \* breadth \* height;

}

int main() {

Box b;

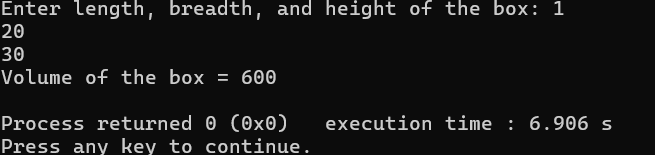
b.read();

cout << "Volume of the box = " << b.calculateVolume() << endl;

return 0;

}

Output.



# DISCUSSION

In this lab session, we learned about the use of class and objects in C++ programming. Along with the class and objects, the use of constructors and the significance of using constructors and destructors were discussed by practicing some problems by using them. We learned different types of constructor copy constructor, parameterized constructors and default constructors and the use of destructors to clean up the resources when object is destroyed.

# CONCLUSION

Hence from this session, we used the concepts of classes, objects, constructors and destructors. Conclusively, we learnt how to use these core concepts of C++ to solve various problems.