**DAY 6 LAB**

**1.PROGRAM**

#include <iostream>

#include <string>

using namespace std;

class Student

{

public:

string name;

int rollNo;

char grade;

Student(string n, int r, char g)

{

name = n;

rollNo = r;

grade = g;

}

void display()

{

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

cout << "Roll No: " << rollNo << endl;

cout << "Grade: " << grade << endl;

cout << endl;

}

};

int main()

{

Student students[3] = {

Student("venu", 1, 'A'),

Student("malli", 2, 'B'),

Student("Bobbi", 3, 'C')

};

cout << "Student Details:" << endl;

for (int i = 0; i < 3; i++)

{

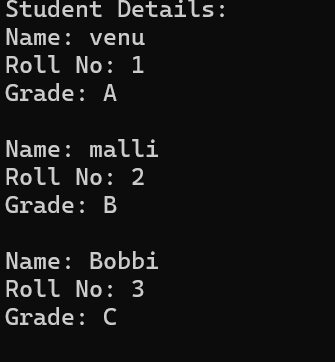
students[i].display();

}

return 0;

}

**OUTPUT**



**2.PROGRAM**

#include <iostream>

#include <string>

using namespace std;

class MyClass

{

public:

int myInt;

double myDouble;

string myString;

MyClass(int i, double d, string s)

{

myInt = i;

myDouble = d;

myString = s;

}

void display()

{

cout << "myInt: " << myInt << endl;

cout << "myDouble: " << myDouble << endl;

cout << "myString: " << myString << endl;

}

};

int main()

{

MyClass obj(10, 3.14, "Hello, World!");

obj.display();

return 0;

}

**OUTPUT**



**3.PROGRAM**

#include <iostream>

using namespace std;

inline double areaCircle(double radius)

{

return 3.14159 \* radius \* radius;

}

inline double areaRectangle(double length, double width)

{

return length \* width;

}

inline double areaSquare(double side)

{

return side \* side;

}

inline double areaTriangle(double base, double height)

{

return 0.5 \* base \* height;

}

int main()

{

double radius, length, width, side, base, height;

cout << "Enter the radius of the circle: ";

cin >> radius;

cout << "Area of the circle: " << areaCircle(radius) << endl;

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

cin >> length >> width;

cout << "Area of the rectangle: " << areaRectangle(length, width) << endl;

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

cin >> side;

cout << "Area of the square: " << areaSquare(side) << endl;

cout << "Enter the base and height of the triangle: ";

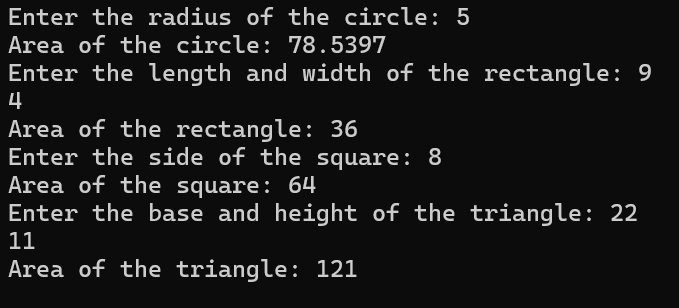
cin >> base >> height;

cout << "Area of the triangle: " << areaTriangle(base, height) << endl;

return 0;

}

**OUTPUT**



**4.PROGRAM**

#include <iostream>

using namespace std;

inline double add(double a, double b)

{

return a + b;

}

inline double subtract(double a, double b)

{

return a - b;

}

inline double multiply(double a, double b)

{

return a \* b;

}

inline double divide(double a, double b)

{

return a / b;

}

inline int modulus(int a, int b)

{

return a % b;

}

int main()

{

double num1, num2;

int intNum1, intNum2;

cout << "Enter two numbers for arithmetic operations: ";

cin >> num1 >> num2;

cout << "Enter two integers for modulus operation: ";

cin >> intNum1 >> intNum2;

cout << "Addition: " << add(num1, num2) << endl;

cout << "Subtraction: " << subtract(num1, num2) << endl;

cout << "Multiplication: " << multiply(num1, num2) << endl;

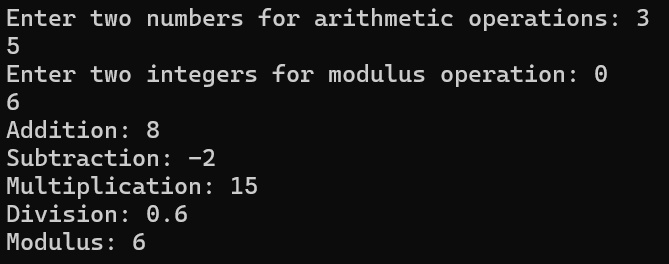
cout << "Division: " << divide(num1, num2) << endl;

cout << "Modulus: " << (intNum1, intNum2) << endl;

return 0;

}

**OUTPUT**



**5.PROGRAM**

#include <iostream>

using namespace std;

void swap(int num1, int num2)

{

int temp = num1;

num1 = num2;

num2 = temp;

}

int main()

{

int num1, num2;

cout << "Enter two numbers to swap: ";

cin >> num1 >> num2;

cout << "Before swapping: num1 = " << num1 << ", num2 = " << num2 << endl;

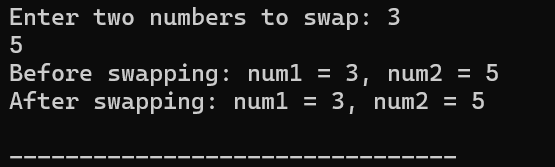
swap(num1, num2);

cout << "After swapping: num1 = " << num1 << ", num2 = " << num2 << endl;

return 0;

}

**OUTPUT**



**6.PROGRAM**

#include <iostream>

#include <string>

using namespace std;

class Employee

{

protected:

string empName;

int empId;

double basicSalary;

public:

Employee(string name, int id, double salary)

{

empName = name;

empId = id;

basicSalary = salary;

}

};

class Payroll : public Employee

{

private:

double grossPay;

double netPay;

public:

Payroll(string name, int id, double salary) : Employee(name, id, salary)

{

grossPay = basicSalary;

netPay = grossPay;

}

void display()

{

cout << "Employee Name: " << empName << endl;

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

cout << "Basic Salary: $" << basicSalary << endl;

cout << "Gross Pay: $" << grossPay << endl;

cout << "Net Pay: $" << netPay << endl;

cout << endl;

}

};

int main()

{

Payroll employee1("Prasanth", 1, 400700);

Payroll employee2("Alleebaba", 2, 2789239);

Payroll employee3("Bhasksr", 3, 150000);

employee1.display();

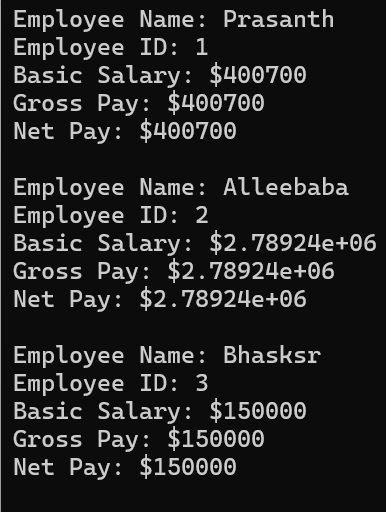
employee2.display();

employee3.display();

return 0;

}

**OUTPUT**



**7.PROGRAM**

#include <iostream>

#include <cmath>

using namespace std;

class InterestCalculator

{

public:

static double simpleInterest(double principal, double rate, double time)

{

return (principal \* rate \* time) / 100.0;

}

static double compoundInterest(double principal, double rate, double time, int n = 1)

{

return principal \* pow((1 + rate / (100.0 \* n)), n \* time) - principal;

}

};

int main()

{

double principal, rate, time;

cout << "Enter principal amount: ";

cin >> principal;

cout << "Enter interest rate (in percentage): ";

cin >> rate;

cout << "Enter time period (in years): ";

cin >> time;

double simple = InterestCalculator::simpleInterest(principal, rate, time);

cout << "Simple Interest: " << simple << endl;

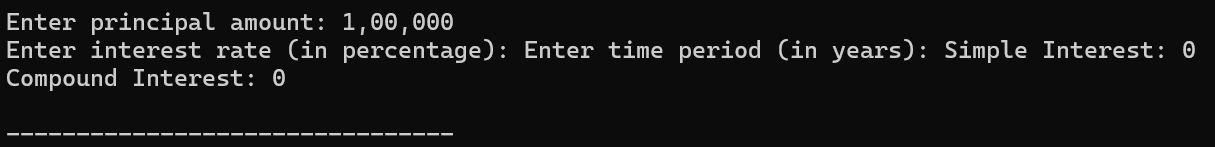
double compound = InterestCalculator::compoundInterest(principal, rate, time);

cout << "Compound Interest: " << compound << endl;

return 0;

}

**OUTPUT**



**8.PROGRAM**

#include <iostream>

using namespace std;

class ObjectCounter

{

private:

static int objectCount;

public:

ObjectCounter()

{

objectCount++;

}

~ObjectCounter() {

objectCount--;

}

static int getObjectCount()

{

return objectCount;

}

};

int ObjectCounter::objectCount = 0;

int main()

{

{

ObjectCounter obj1;

cout << "Objects created: " << ObjectCounter::getObjectCount() << endl;

}

cout << "Objects remaining after first block: " << ObjectCounter::getObjectCount() << endl;

{

ObjectCounter obj2;

ObjectCounter obj3;

cout << "Objects created: " << ObjectCounter::getObjectCount() << endl;

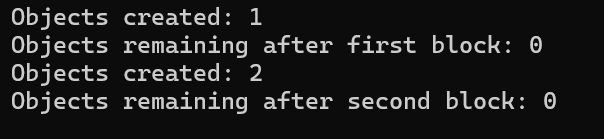
}

cout << "Objects remaining after second block: " << ObjectCounter::getObjectCount() << endl;

return 0;

}

**OUTPUT**



**9.PROGRAM**

#include <iostream>

using namespace std;

class Shape

{

public:

virtual double area() = 0;

};

class Rectangle : public Shape

{

protected:

double length;

double width;

public:

Rectangle(double l, double w) : length(l), width(w) {}

double area() override

{

return length \* width;

}

};

class Cuboid : public Rectangle

{

private:

double height;

public:

Cuboid(double l, double w, double h) : Rectangle(l, w), height(h) {}

double volume()

{

return length \* width \* height;

}

double totalSurfaceArea()

{

return 2 \* (length \* width + length \* height + width \* height);

}

};

int main()

{

double length, width, height;

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

cin >> length >> width >> height;

Cuboid cuboid(length, width, height);

cout << "Area of the cuboid: " << cuboid.area() << endl;

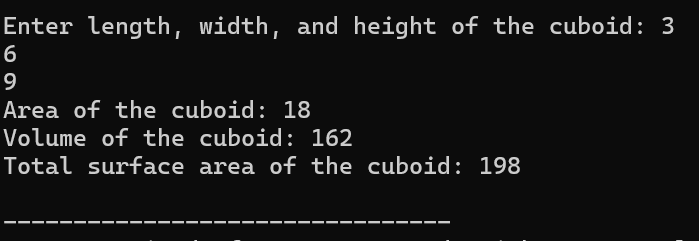
cout << "Volume of the cuboid: " << cuboid.volume() << endl;

cout << "Total surface area of the cuboid: " << cuboid.totalSurfaceArea() << endl;

return 0;

}

**OUTPUT**



**10.PROGRAM**

#include <iostream>

#include <cmath>

using namespace std;

class Circle

{

protected:

double radius;

public:

Circle(double r) : radius(r) {}

double area()

{

return M\_PI \* radius \* radius;

}

};

class Sphere : public Circle

{

public:

Sphere(double r) : Circle(r) {}

double volume()

{

return (4.0 / 3.0) \* M\_PI \* pow(radius, 3);

}

};

class Cylinder : public Sphere

{

private:

double height;

public:

Cylinder(double r, double h) : Sphere(r), height(h) {}

double volume()

{

return Circle::area() \* height;

}

};

int main()

{

double radius, height;

cout << "Enter the radius of the circle: ";

cin >> radius;

cout << "Enter the height of the cylinder: ";

cin >> height;

Cylinder cylinder(radius, height);

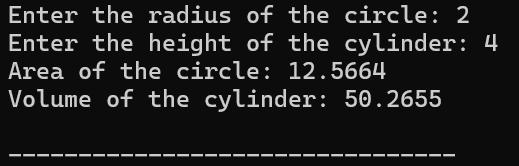
cout << "Area of the circle: " << cylinder.area() << endl;

cout << "Volume of the cylinder: " << cylinder.volume() << endl;

return 0;

}

**OUTPUT**



**11.PROGRAM**

#include <iostream>

#include <vector>

#include <iomanip>

using namespace std;

class Item

{

private:

int codeNo;

double price;

public:

Item(int code, double p) : codeNo(code), price(p) {}

double getPrice() const

{

return price;

}

void display() const

{

cout << "Code No: " << codeNo << ", Price: $" << fixed << setprecision(2) << price << endl;

}

};

class ShoppingList

{

private:

vector<Item> items;

public:

void addItem(const Item& item)

{

items.push\_back(item);

}

void deleteItem(int code)

{

for (auto it = items.begin(); it != items.end(); ++it)

{

if (it->getPrice() == code)

{

items.erase(it);

break;

}

}

}

void printTotalValue() const

{

double total = 0.0;

for (const auto& item : items)

{

total += item.getPrice();

}

cout << "Total value of the order: $" << fixed << setprecision(2) << total << endl;

}

void displayList() const

{

cout << "Shopping List:" << endl;

for (const auto& item : items)

{

item.display();

}

}

};

int main()

{

ShoppingList list;

list.addItem(Item(101, 10.99));

list.addItem(Item(102, 5.99));

list.addItem(Item(103, 7.50));

list.displayList();

list.printTotalValue();

list.deleteItem(102);

cout << "\nAfter deleting an item:" << endl;

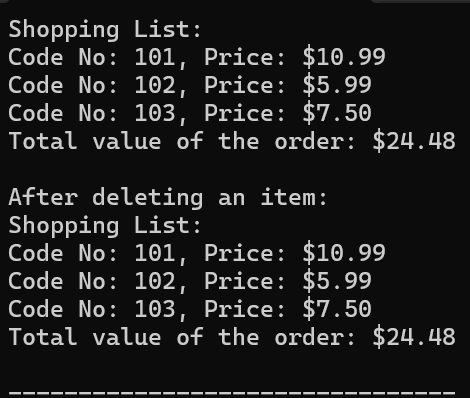
list.displayList();

list.printTotalValue();

return 0;

}

**OUTPUT**



**12.PROGRAM**

#include <iostream>

using namespace std;

class Shape

{

protected:

double height;

double width;

public:

Shape(double h, double w) : height(h), width(w) {}

virtual double area() const = 0;

virtual double perimeter() const = 0;

};

class Rectangle : public Shape

{

public:

Rectangle(double h, double w) : Shape(h, w) {}

double area() const override

{

return height \* width;

}

double perimeter() const override

{

return 2 \* (height + width);

}

};

class Triangle : public Shape

{

private:

double side1;

double side2;

double side3;

public:

Triangle(double h, double w, double s1, double s2, double s3) : Shape(h, w), side1(s1), side2(s2), side3(s3) {}

double area() const override

{

double s = (side1 + side2 + side3) / 2;

return (s \* (s - side1) \* (s - side2) \* (s - side3));

}

double perimeter() const override

{

return side1 + side2 + side3;

}

};

int main()

{

double height, width, side1, side2, side3;

cout << "Enter height and width of rectangle: ";

cin >> height >> width;

Rectangle rectangle(height, width);

cout << "Enter lengths of three sides of triangle: ";

cin >> side1 >> side2 >> side3;

Triangle triangle(height, width, side1, side2, side3);

cout << "\nRectangle:" << endl;

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

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

cout << "\nTriangle:" << endl;

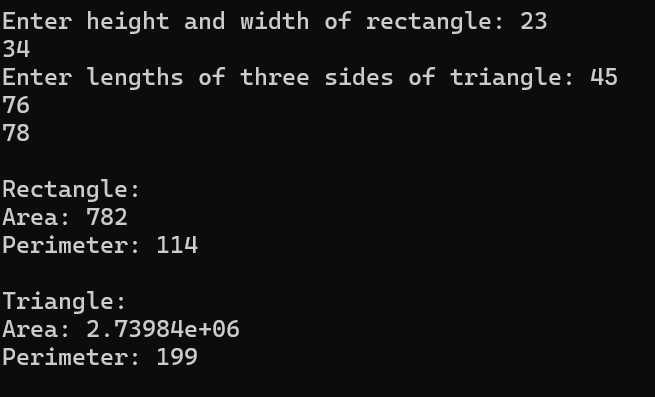
cout << "Area: " << triangle.area() << endl;

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

return 0;

}

**OUTPUT**



**13.PROGRAM**

#include <iostream>

#include <string>

using namespace std;

class Employee

{

protected:

string empName;

int empId;

string address;

string mailId;

string mobileNo;

public:

Employee(string name, int id, string addr, string mail, string mobile)

: empName(name), empId(id), address(addr), mailId(mail), mobileNo(mobile) {}

virtual double calculateBasicPay() const = 0;

void generatePaySlip() const

{

double basicPay = calculateBasicPay();

double da = 0.97 \* basicPay;

double hra = 0.10 \* basicPay;

double pf = 0.12 \* basicPay;

double staffClubFund = 0.001 \* basicPay;

double grossSalary = basicPay + da + hra;

double netSalary = grossSalary - pf - staffClubFund;

cout << "\nPay Slip" << endl;

cout << "Employee Name: " << empName << endl;

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

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

cout << "Mail ID: " << mailId << endl;

cout << "Mobile No: " << mobileNo << endl;

cout << "Basic Pay: $" << basicPay << endl;

cout << "Dearness Allowance (DA): $" << da << endl;

cout << "House Rent Allowance (HRA): $" << hra << endl;

cout << "Provident Fund (PF): $" << pf << endl;

cout << "Staff Club Fund: $" << staffClubFund << endl;

cout << "Gross Salary: $" << grossSalary << endl;

cout << "Net Salary: $" << netSalary << endl;

}

};

class Programmer : public Employee

{

private:

double basicPay;

public:

Programmer(string name, int id, string addr, string mail, string mobile, double bp)

: Employee(name, id, addr, mail, mobile), basicPay(bp) {}

double calculateBasicPay() const override

{

return basicPay;

}

};

class AssistantProfessor : public Employee

{

private:

double basicPay;

public:

AssistantProfessor(string name, int id, string addr, string mail, string mobile, double bp)

: Employee(name, id, addr, mail, mobile), basicPay(bp) {}

double calculateBasicPay() const override

{

return basicPay;

}

};

class AssociateProfessor : public Employee

{

private:

double basicPay;

public:

AssociateProfessor(string name, int id, string addr, string mail, string mobile, double bp)

: Employee(name, id, addr, mail, mobile), basicPay(bp) {}

double calculateBasicPay() const override

{

return basicPay;

}

};

class Professor : public Employee

{

private:

double basicPay;

public:

Professor(string name, int id, string addr, string mail, string mobile, double bp)

: Employee(name, id, addr, mail, mobile), basicPay(bp) {}

double calculateBasicPay() const override

{

return basicPay;

}

};

int main()

{

Programmer programmer("Prasanth", 101, "123 Main St", "john@example.com", "1234567890", 5000);

AssistantProfessor assistantProfessor("Alleebaba", 102, "456 Oak Ave", "alice@example.com", "9876543210", 6000);

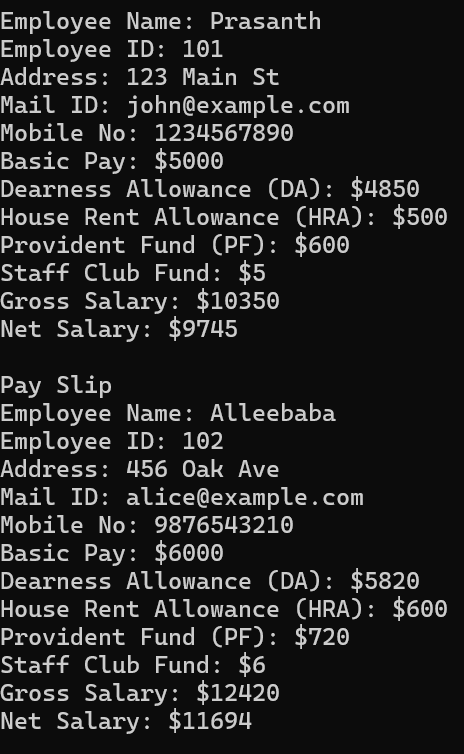
programmer.generatePaySlip();

assistantProfessor.generatePaySlip();

return 0;

}

**Output**



**14.PROGRAM**

#include <iostream>

using namespace std;

int main()

{

int marks1, marks2, marks3, marks4;

cout << "Enter marks for subject 1: ";

cin >> marks1;

cout << "Enter marks for subject 2: ";

cin >> marks2;

cout << "Enter marks for subject 3: ";

cin >> marks3;

cout << "Enter marks for subject 4: ";

cin >> marks4;

int totalMarks = marks1 + marks2 + marks3 + marks4;

double aggregate = totalMarks / 4.0;

cout << "\nTotal Marks: " << totalMarks << endl;

cout << "Aggregate: " << aggregate << "%" << endl;

cout << "\nGrade: ";

if (aggregate > 75)

{

cout << "Distinction" << endl;

} else if (aggregate >= 60 && aggregate < 75)

{

cout << "First Division" << endl;

} else if (aggregate >= 50 && aggregate < 60)

{

cout << "Second Division" << endl;

} else if (aggregate >= 40 && aggregate < 50)

{

cout << "Third Division" << endl;

} else {

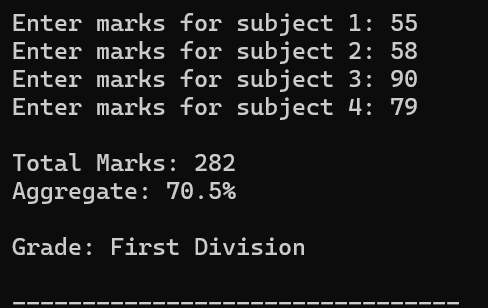
cout << "Fail" << endl;

}

return 0;

}

**OUTPUT**



**15.PROGRAM**

#include <iostream>

using namespace std;

class Person

{

protected:

string name;

public:

void getData()

{

cout << "Enter name: ";

cin >> name;

}

void displayData() const

{

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

}

virtual double bonus() const = 0;

};

class Admin : public Person

{

protected:

double salary;

public:

void getData()

{

Person::getData();

cout << "Enter salary: ";

cin >> salary;

}

void displayData() const

{

Person::displayData();

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

}

double bonus() const override

{

return salary + 1000;

}

};

class Account : public Person

{

protected:

double salary;

public:

void getData()

{

Person::getData();

cout << "Enter salary: ";

cin >> salary;

}

void displayData() const

{

Person::displayData();

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

}

double bonus() const override

{

return salary + 2000;

}

};

class Master : public Admin, public Account

{

public:

void getData()

{

Admin::getData();

Account::getData();

}

void displayData() const

{

Admin::displayData();

Account::displayData();

}

double bonus() const override

{

return (Admin::bonus() + Account::bonus()) / 2.0;

}

};

int main()

{

Master employee;

employee.getData();

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

employee.displayData();

cout << "\nBonus: $" << employee.bonus() << endl;

return 0;

}

**OUTPUT**

