**Program #01:**

#include <iostream>

#include <string>

using namespace std;

class Publication

{

protected:

string title;

float price;

public:

Publication() : title(""), price(0.0f) {}

virtual void getdata()

{

cout << "Enter title: ";

getline(cin, title);

cout << "Enter price: ";

cin >> price;

cin.ignore();

}

virtual void putdata() const

{

cout << "Title: " << title << endl;

cout << "Price: $" << price << endl;

}

};

class Book : public Publication {

private:

int pageCount;

public:

Book() : pageCount(0) {}

void getdata() override {

Publication::getdata();

cout << "Enter page count: ";

cin >> pageCount;

cin.ignore();

}

void putdata() const override {

Publication::putdata();

cout << "Page count: " << pageCount << endl;

}

};

class Tape : public Publication {

private:

float playingTime;

public:

Tape() : playingTime(0.0f) {}

void getdata() override {

Publication::getdata();

cout << "Enter playing time (in minutes): ";

cin >> playingTime;

cin.ignore();

}

void putdata() const override {

Publication::putdata();

cout << "Playing time: " << playingTime << " minutes" << endl;

}

};

int main() {

Book book;

Tape tape;

cout << "Enter data for book:" << endl;

book.getdata();

cout << "Enter data for tape:" << endl;

tape.getdata();

cout << "Displaying book data:" << endl;

book.putdata();

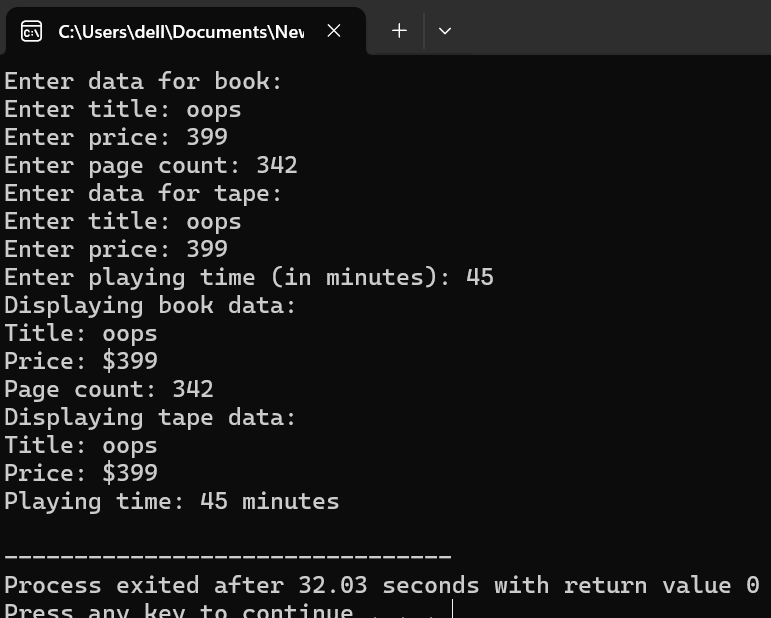
cout << "Displaying tape data:" << endl;

tape.putdata();

return 0;

}

**Output:**

****

**Program# 02:**

#include <iostream>

#include <string>

using namespace std;

class Publication {

protected:

string title;

float price;

public:

Publication() : title(""), price(0.0f) {}

virtual void getdata() {

cout << "Enter title: ";

getline(cin, title);

cout << "Enter price: ";

cin >> price;

cin.ignore();

}

virtual void putdata() const {

cout << "Title: " << title << endl;

cout << "Price: $" << price << endl;

}

};

class Sales {

protected:

float sales[3];

public:

Sales() {

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

sales[i] = 0.0f;

}

}

void getdata() {

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

cout << "Enter sales for month " << i + 1 << ": ";

cin >> sales[i];

}

}

void putdata() const {

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

cout << "Sales for month " << i + 1 << ": $" << sales[i] << endl;

}

}

};

class Book : public Publication, public Sales {

private:

int pageCount;

public:

Book() : pageCount(0) {}

void getdata() override {

Publication::getdata();

cout << "Enter page count: ";

cin >> pageCount;

cin.ignore();

Sales::getdata();

}

void putdata() const override {

Publication::putdata();

cout << "Page count: " << pageCount << endl;

Sales::putdata();

}

};

class Tape : public Publication, public Sales {

private:

float playingTime;

public:

Tape() : playingTime(0.0f) {}

void getdata() override {

Publication::getdata();

cout << "Enter playing time (in minutes): ";

cin >> playingTime;

cin.ignore(); // To consume the newline character left in the buffer

Sales::getdata();

}

void putdata() const override {

Publication::putdata();

cout << "Playing time: " << playingTime << " minutes" << endl;

Sales::putdata();

}

};

int main() {

Book book;

Tape tape;

cout << "Enter data for book:" << endl;

book.getdata();

cout << "\nEnter data for tape:" << endl;

tape.getdata();

cout << "\nDisplaying book data:" << endl;

book.putdata();

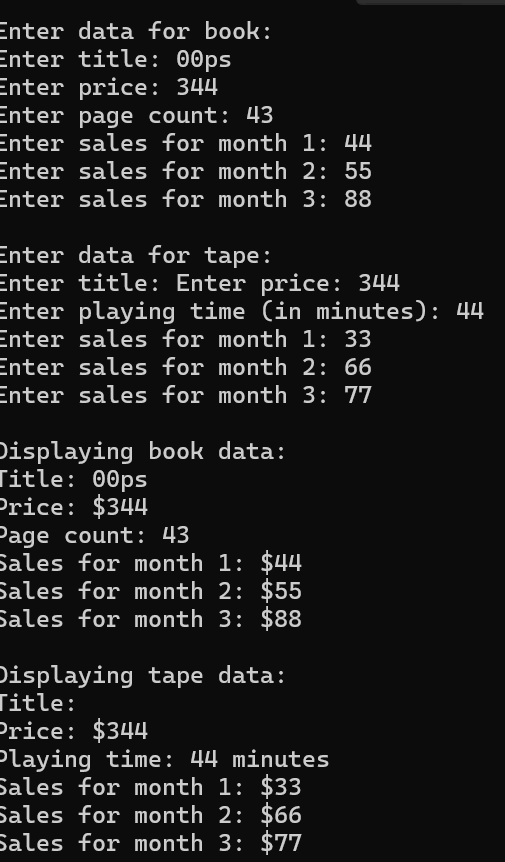
cout << "\nDisplaying tape data:" << endl;

tape.putdata();

return 0;

}

**Output:**



**Programe # 03:**

#include <iostream>

#include <string>

using namespace std;

class Publication {

protected:

string title;

float price;

public:

Publication() : title(""), price(0.0f) {}

virtual void getdata() {

cout << "Enter title: ";

getline(cin, title);

cout << "Enter price: ";

cin >> price;

cin.ignore();

}

virtual void putdata() const {

cout << "Title: " << title << endl;

cout << "Price: $" << price << endl;

}

};

class Sales {

protected:

float sales[3];

public:

Sales() {

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

sales[i] = 0.0f;

}

}

void getdata() {

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

cout << "Enter sales for month " << i + 1 << ": ";

cin >> sales[i];

}

}

void putdata() const {

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

cout << "Sales for month " << i + 1 << ": $" << sales[i] << endl;

}

}

};

class Book : public Publication, public Sales {

private:

int pageCount;

public:

Book() : pageCount(0) {}

void getdata() override {

Publication::getdata();

cout << "Enter page count: ";

cin >> pageCount;

cin.ignore();

Sales::getdata();

}

void putdata() const override {

Publication::putdata();

cout << "Page count: " << pageCount << endl;

Sales::putdata();

}

};

class Tape : public Publication, public Sales {

private:

float playingTime;

public:

Tape() : playingTime(0.0f) {}

void getdata() override {

Publication::getdata();

cout << "Enter playing time (in minutes): ";

cin >> playingTime;

cin.ignore();

Sales::getdata();

}

void putdata() const override {

Publication::putdata();

cout << "Playing time: " << playingTime << " minutes" << endl;

Sales::putdata();

}

};

class Disk : public Publication, public Sales {

private:

enum DiskType { CD, DVD } diskType;

public:

Disk() : diskType(CD) {}

void getdata() override {

Publication::getdata();

char type;

cout << "Enter disk type (c for CD, d for DVD): ";

cin >> type;

cin.ignore(); // To consume the newline character left in the buffer

diskType = (type == 'c' || type == 'C') ? CD : DVD;

Sales::getdata();

}

void putdata() const override {

Publication::putdata();

cout << "Disk type: " << (diskType == CD ? "CD" : "DVD") << endl;

Sales::putdata();

}

};

int main() {

Book book;

Tape tape;

Disk disk;

cout << "Enter data for book:" << endl;

book.getdata();

cout << "\nEnter data for tape:" << endl;

tape.getdata();

cout << "\nEnter data for disk:" << endl;

disk.getdata();

cout << "\nDisplaying book data:" << endl;

book.putdata();

cout << "\nDisplaying tape data:" << endl;

tape.putdata();

cout << "\nDisplaying disk data:" << endl;

disk.putdata();

return 0;

}

**Program #04:**

#include <iostream>

#include <string>

using namespace std;

class Employee {

protected:

string name;

unsigned long number;

public:

void getdata() {

cout << "Enter name: ";

getline(cin, name);

cout << "Enter number: ";

cin >> number;

cin.ignore();

}

void putdata() const {

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

cout << "Number: " << number << endl;

}

};

class Employee2 : public Employee {

public:

enum Period { HOURLY, WEEKLY, MONTHLY };

private:

double compensation;

Period period;

public:

void getdata() {

Employee::getdata();

cout << "Enter compensation: ";

cin >> compensation;

cin.ignore();

char periodInput;

cout << "Enter period (h for hourly, w for weekly, m for monthly): ";

cin >> periodInput;

cin.ignore();

switch (periodInput) {

case 'h':

case 'H':

period = HOURLY;

break;

case 'w':

case 'W':

period = WEEKLY;

break;

case 'm':

case 'M':

period = MONTHLY;

break;

default:

cout << "Invalid input. Defaulting to hourly." << endl;

period = HOURLY;

}

}

void putdata() const {

Employee::putdata();

cout << "Compensation: " << compensation << endl;

cout << "Period: " << (period == HOURLY ? "Hourly" : period == WEEKLY ? "Weekly" : "Monthly") << endl;

}

};

class Manager : public Employee2 {};

class Scientist : public Employee2 {};

class Laborer : public Employee2 {};

int main() {

Manager m;

Scientist s;

Laborer l;

cout << "Enter data for manager:" << endl;

m.getdata();

cout << "\nEnter data for scientist:" << endl;

s.getdata();

cout << "\nEnter data for laborer:" << endl;

l.getdata();

cout << "\nDisplaying manager data:" << endl;

m.putdata();

cout << "\nDisplaying scientist data:" << endl;

s.putdata();

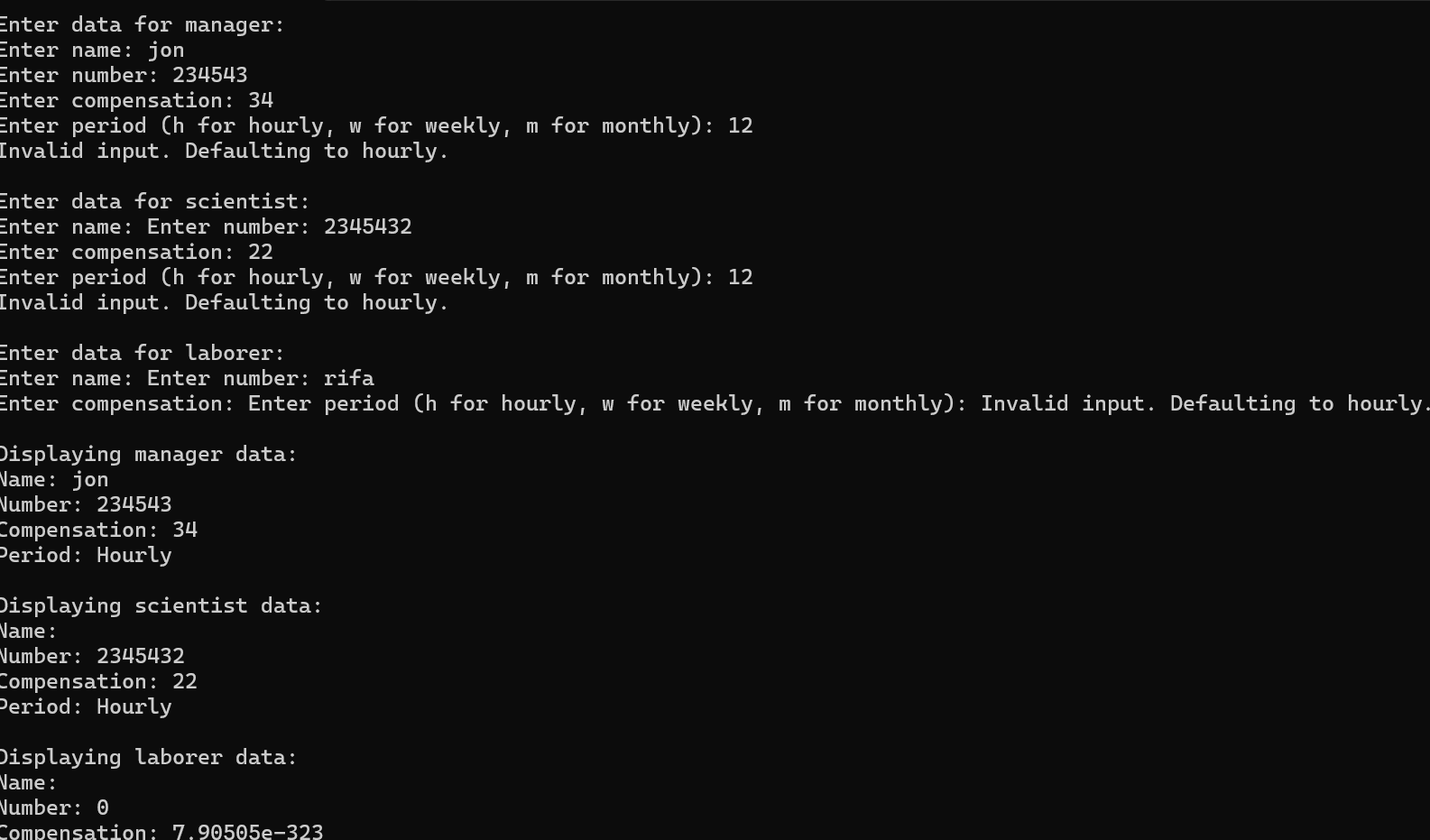
cout << "\nDisplaying laborer data:" << endl;

l.putdata();

return 0;

}

**Output:**



**Program #05:**

#include <iostream>

#include <string>

using namespace std;

class Employee {

protected:

string name;

int employeeID;

string department;

public:

void setName(const string& n) {

name = n;

}

string getName() const {

return name;

}

void setEmployeeID(int id) {

employeeID = id;

}

int getEmployeeID() const {

return employeeID;

}

void setDepartment(const string& dept) {

department = dept;

}

string getDepartment() const {

return department;

}

virtual double calculatePay() const = 0;

};

class SalariedEmployee : public Employee {

private:

double annualSalary;

public:

void setAnnualSalary(double salary) {

annualSalary = salary;

}

double getAnnualSalary() const {

return annualSalary;

}

double calculatePay() const override {

return annualSalary / 12.0;

}

};

class CommissionEmployee : public Employee {

private:

double sales;

double commissionRate;

public:

void setSales(double s) {

sales = s;

}

double getSales() const {

return sales;

}

void setCommissionRate(double rate) {

commissionRate = rate;

}

double getCommissionRate() const {

return commissionRate;

}

double calculatePay() const override {

return sales \* commissionRate;

}

};

int main() {

SalariedEmployee salariedEmp;

salariedEmp.setName("John Doe");

salariedEmp.setEmployeeID(101);

salariedEmp.setDepartment("Engineering");

salariedEmp.setAnnualSalary(60000);

CommissionEmployee commissionEmp;

commissionEmp.setName("Jane Smith");

commissionEmp.setEmployeeID(102);

commissionEmp.setDepartment("Sales");

commissionEmp.setSales(150000);

commissionEmp.setCommissionRate(0.1);

cout << "Salaried Employee:" << endl;

cout << "Name: " << salariedEmp.getName() << endl;

cout << "Employee ID: " << salariedEmp.getEmployeeID() << endl;

cout << "Department: " << salariedEmp.getDepartment() << endl;

cout << "Annual Salary: $" << salariedEmp.getAnnualSalary() << endl;

cout << "Monthly Pay: $" << salariedEmp.calculatePay() << endl;

cout << "\nCommission Employee:" << endl;

cout << "Name: " << commissionEmp.getName() << endl;

cout << "Employee ID: " << commissionEmp.getEmployeeID() << endl;

cout << "Department: " << commissionEmp.getDepartment() << endl;

cout << "Sales: $" << commissionEmp.getSales() << endl;

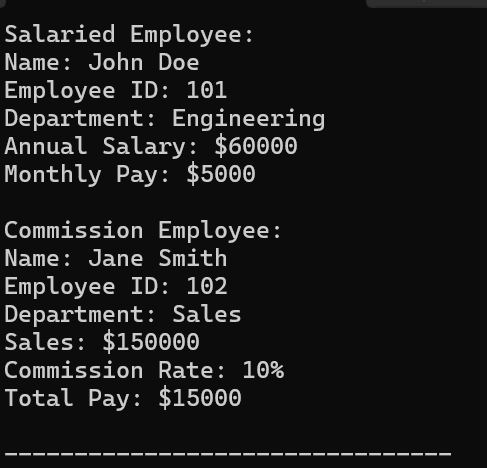
cout << "Commission Rate: " << commissionEmp.getCommissionRate() \* 100 << "%" << endl;

cout << "Total Pay: $" << commissionEmp.calculatePay() << endl;

return 0;

}

**Output:**



**Program #06:**

#include <iostream>

#include <string>

#include <cmath>

using namespace std;

class Shape {

protected:

string color;

public:

Shape(const string& c) : color(c) {}

void printColor() const {

cout << "Color: " << color << endl;

}

};

class Circle : public Shape {

private:

double radius;

public:

Circle(const string& c, double r) : Shape(c), radius(r) {}

double calculateArea() const {

return M\_PI \* radius \* radius;

}

void printArea() const {

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

}

};

class Rectangle : public Shape {

private:

double length;

double width;

public:

Rectangle(const string& c, double l, double w) : Shape(c), length(l), width(w) {}

double calculateArea() const {

return length \* width;

}

void printArea() const {

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

}

};

int main() {

// Create a Circle object

Circle circle("Red", 5.0);

cout << "Circle details:" << endl;

circle.printColor();

circle.printArea();

// Create a Rectangle object

Rectangle rectangle("Blue", 4.0, 6.0);

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

rectangle.printColor();

rectangle.printArea();

return 0;

}

**Output:**

