**OBJECT ORIENTED PROGRAMMING**

**HOME WORK**

**TOPIC:INHERITANCE**

**TASK O1:**

Making a class parent class (user) which inherited by child class( student,teacher)

**CODE:**

#include <iostream>

#include <string>

using namespace std;

// Parent class

class User {

protected:

string firstName;

string lastName;

string phone;

string address;

string NIC; // National Identity Card

string emergencyContact;

public:

User(string firstName, string phone, string address, string NIC, string emergencyContact)

: firstName(firstName), phone(phone), address(address), NIC(NIC), emergencyContact(emergencyContact) {}

virtual void displayInfo() {

cout << "Name: " << firstName << "\n"

<< "Phone: " << phone << "\n"

<< "Address: " << address << "\n"

<< "NIC: " << NIC << "\n”

<< "Emergency Contact: " << emergencyContact << endl; }};

// Child class Student inheriting from User using public inheritance

class Student : public User {

private:

string rollNumber;

string seatNumber;

string batchEnrollment;

public:

Student(string firstName, string phone, string address, string NIC, string emergencyContact,

string rollNumber, string seatNumber, string batchEnrollment)

: User(firstName, phone, address, NIC, emergencyContact), rollNumber(rollNumber), seatNumber(seatNumber), batchEnrollment(batchEnrollment) {}

void displayInfo() override {

User::displayInfo();

cout << "Roll Number: " << rollNumber << "\n"

<< "Seat Number: " << seatNumber << "\n"

<< "Batch Enrollment: " << batchEnrollment << endl; }};

class Teacher : protected User {

private:

string batchAdvisor;

string department;

string courseLoad;

public:

Teacher(string firstName, string phone, string address, string NIC, string emergencyContact,

string batchAdvisor, string department, string courseLoad)

: User(firstName, phone, address, NIC, emergencyContact), batchAdvisor(batchAdvisor), department(department), courseLoad(courseLoad) {

void displayInfo() {

User::displayInfo();

cout << "Batch Advisor: " << batchAdvisor << "\n"

<< "Department: " << department << "\n"

<< "Course Load: " << courseLoad << endl; }

void showName() {

cout << "Teacher's name: " << firstName << " " << lastName << endl;}};

int main() {

Student student("kainat moin", "03233161627", "karachi", "42401-1563936-6", "03402365783", "48","Seat1", "2023");

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

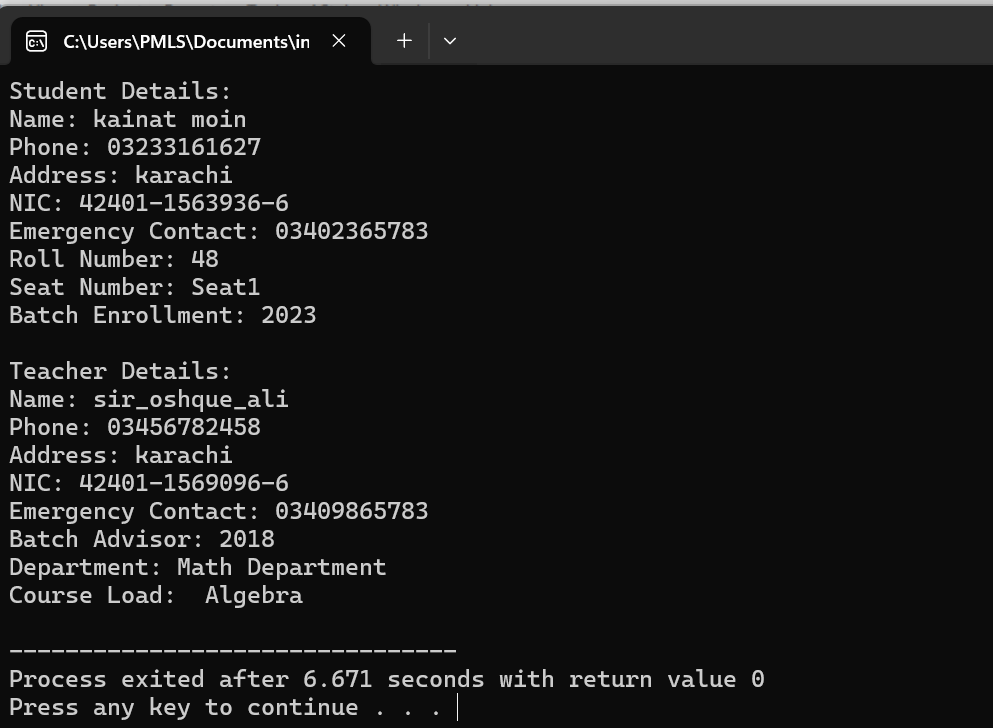
student.displayInfo();

Teacher teacher("sir\_oshque\_ali", "03456782458", "karachi", "42401-1569096-6", "03409865783", "2018", "Math Department", " Algebra");

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

teacher.displayInfo();

return 0;}

**OUTPUT:**

**TASK 02**

Inherited between employ and manager

**1. Using all three access specifiers:**

**CODE:**

#include <iostream>

#include <string>

using namespace std;

// Base class

class Employee {

protected:

string name;

int employeeId;

double salary;

public:

Employee(string name, int employeeId, double salary) : name(name), employeeId(employeeId), salary(salary) {}

virtual void displayDetails() {

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

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

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

};

// Child class inheriting from Employee

class Manager : public Employee {

private:

string department;

public:

// Constructor

Manager(string name, int employeeId, double salary, string department) : Employee(name, employeeId, salary), department(department) {}

/ Overriding the displayDetails() function of the base class

void displayDetails() override {

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

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

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

cout << "Department: " << department << endl;};\

int main() {

// Creating objects of base and child classes

Employee emp("saqib", 1001, 50000);

Manager mgr("Mobin", 2001, 70000, "Marketing");

// Calling displayDetails() function for both objects

cout << "Employee Details:" << endl;

emp.displayDetails();

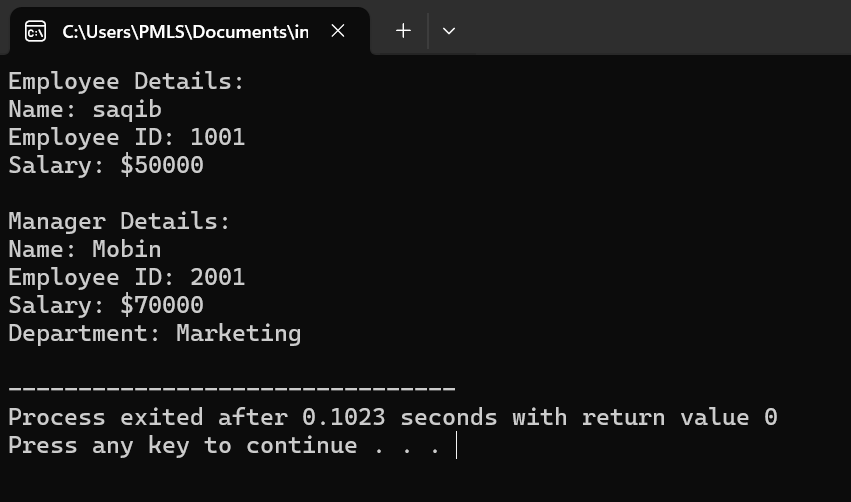
cout << endl;

cout << "Manager Details:" << endl;

mgr.displayDetails();

return 0;}

**OUTPUT:**

****

**Access specifier**

**Public:**

#include <iostream>

using namespace std;

class parent{

public:

int a;

private:

int b;

protected:

int c;

};

class child : public parent{

public:

void in(){

cout<<"Enter the value of a:";

cin>>a;

//cout<<"Enter the value of b:";

//cin>>b;

cout<<"Enter the value of c:";

cin>>c;

}

void out(){

cout<<"a ="<<a<<endl;

//cout<<"b"<<b<<endl;

cout<<"c ="<<c<<endl;

}}; int main (){

child obj;

obj.in();

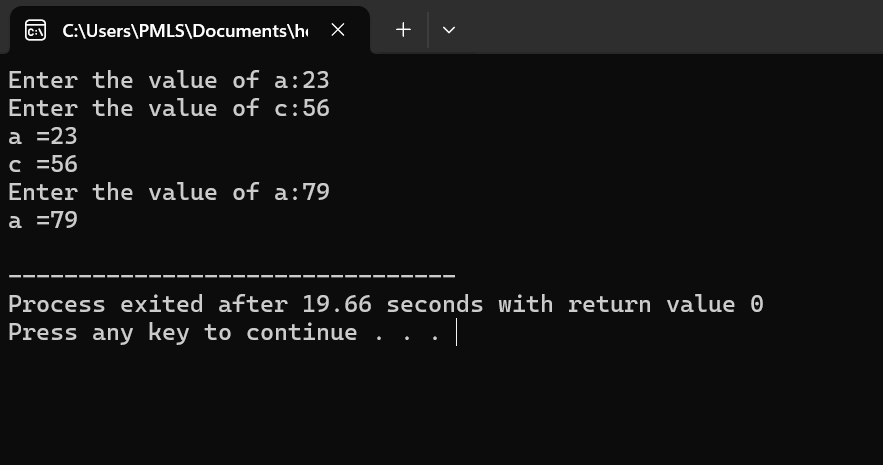
obj.out();

cout<<"Enter the value of a:";

cin>>obj.a;

cout<<"a ="<<obj.a<<endl;

return 0;}

**OUTPUT:**

**Private:**

#include <iostream>

using namespace std;

class parent{

public:

int a;

private:

int b;

protected:

int c;};

class child : private parent{

public:

void in(){cout<<"Enter the value of a:";

cin>>a;

//cout<<"Enter the value of b:";

//cin>>b;

cout<<"Enter the value of c:";

cin>>c;}

void out(){

cout<<"a ="<<a<<endl;

//cout<<"b"<<b<<endl;

cout<<"c ="<<c<<endl; } };

int main (){

child obj;

obj.in();

obj.out();

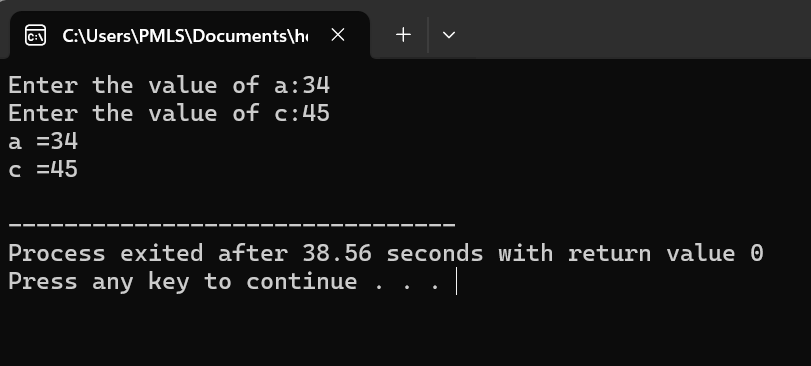
//cout<<"Enter the value of a:";

//cin>>obj .a;

//cout<<"a ="<<obj.a<<endl;

return 0;}

**OUTPUT:**



**Protected:**

#include <iostream>

using namespace std;

class parent{

public:

int a;

private:

int b;

protected:

int c;

};

class child : protected parent{

public:

void in(){

cout<<"Enter the value of a:";

cin>>a;

//cout<<"Enter the value of b:";

//cin>>b;

cout<<"Enter the value of c:";

cin>>c;

}

void out(){

cout<<"a ="<<a<<endl;

//cout<<"b"<<b<<endl;

cout<<"c ="<<c<<endl; } };

int main (){

child obj;

obj.in();

obj.out();

//cout<<"Enter the value of a:";

//cin>>obj.a;

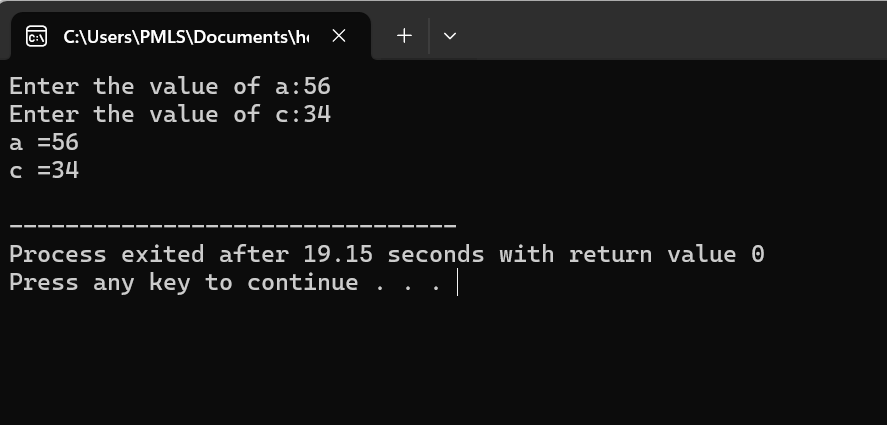
//cout<<"Enter the value of c:";

//cin>>obj.c;

//cout<<"a ="<<obj.a<<endl;

//cout<<"c ="<<obj.c<<endl;

return 0;}

**OUTPUT:**

**Question no 1**

Imagine a publishing company that markets both book and audiocassette versions of its works. Create a class publication that stores the title (a string) and price (type float) of a publication. From this class derive two classes: book, which adds a page count (type int), and tape, which adds a playing time in minutes (type float). Each of these three classes should have a getdata() function to get its data from the user at the keyboard, and a putdata() function to display its data. Write a main() program to test the book and tape classes by creating instances of them, asking the user to fill in data with getdata(), and then displaying the data with putdata().

**CODE:**

#include <iostream>

#include <string>

using namespace std;

class publication {

protected:

string title;

float price;

public:

void getdata() {

cout << "Enter title: ";

getline(cin, title);

cout << "Enter price: ";

cin >> price;

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

}

void putdata() const {

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

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

}

};

class book : public publication {

private:

int pageCount;

public:

void getdata() {

publication::getdata();

cout << "Enter page count: ";

cin >> pageCount;

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

}

void putdata() const {

publication::putdata();

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

}

};

class tape : public publication {

private:

float playingTime;

public:

void getdata() {

publication::getdata();

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

cin >> playingTime;

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

}

void putdata() const {

publication::putdata();

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

}

};

int main() {

book myBook;

tape myTape;

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

myBook.getdata();

cout << endl;

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

myTape.getdata();

cout << endl;

cout << "Details of book:" << endl;

myBook.putdata();

cout << endl;

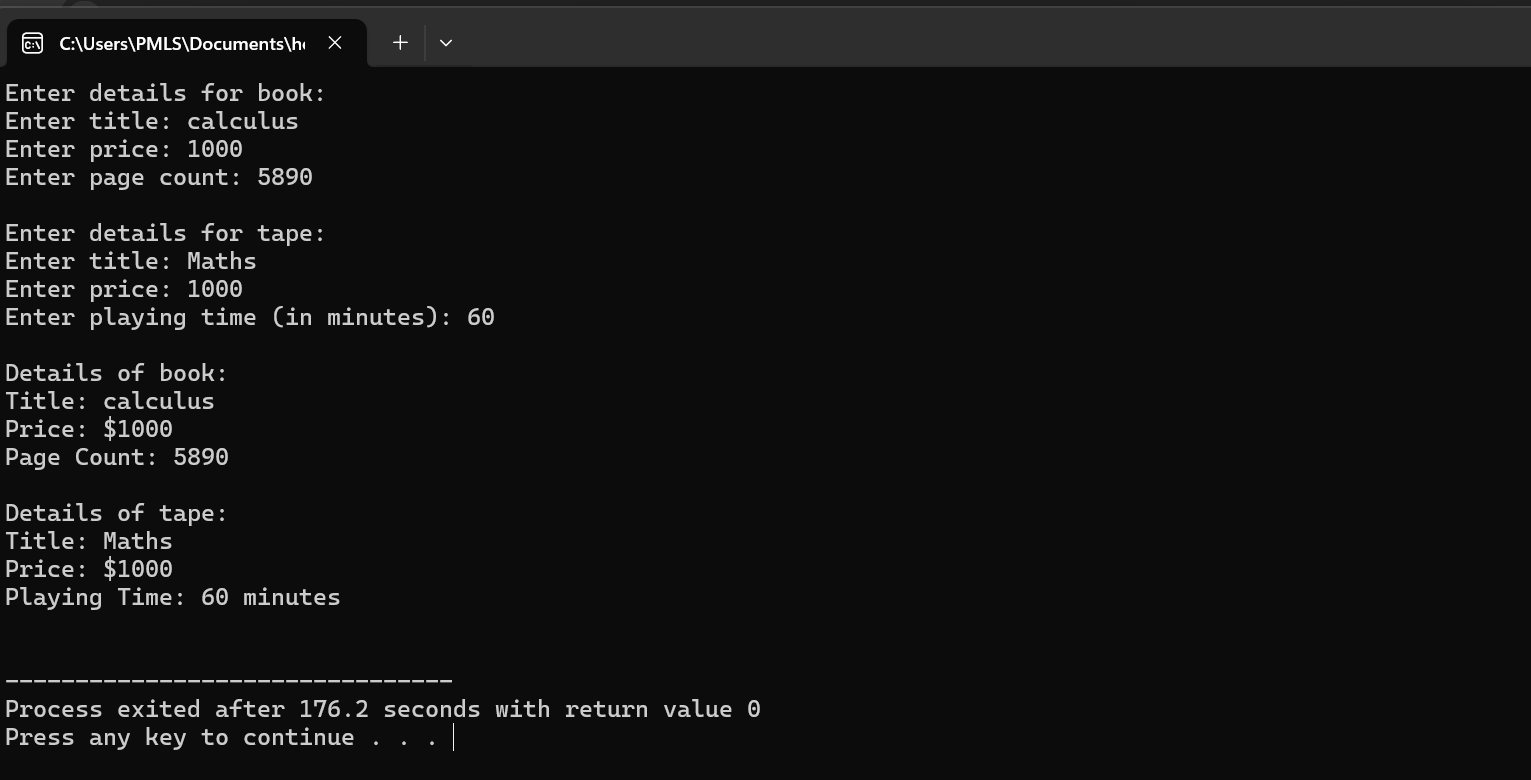
cout << "Details of tape:" << endl;

myTape.putdata();

cout << endl;

return 0;}

**OUTPUT**



**Question no 2**

In a game, there are three kinds of characters: warriors, wizards, and rogues. All characters

have a name and a level. Warriors have additional attributes for strength and defense,

wizards have attributes for mana and spell power, and rogues have attributes for agility and

stealth. The game stores each character's data and allows for collecting and displaying this

information.

After defining C++ class definitions and a main function, which interactively collects and

displays data for various characters(use function overriding), answer the following

questions:

1) Describe how inheritance is used to manage character data in this scenario.

2) Write the output of the program if the input provided by the user for warrior 1 is "Aragorn",

10, 150, 100; for wizard 1 is "Gandalf", 15, 200, 250; and for rogue 1 is "Legolas", 12, 180, 90.

3) Modify the rogue class to include an additional attribute for lock-picking skill and update

the main function to handle this new attribute.

**Answer;**

**1) How Inheritance is Used to Manage Character Data in this Scenario**

Inheritance in C++ allows us to create a base class that holds common attributes and methods for all derived classes, which in this case represent different types of game characters. Each character type (warrior, wizard, rogue) inherits common properties from a base class called Character and adds specific attributes and methods relevant to that type. This helps to avoid code duplication and makes the code more organized and easier to maintain.

**CODE:**

#include <iostream>

#include <string>

using namespace std;

class Character {

protected:

string name;

int level;

public:

virtual void getdata() {

cout << "Enter name: ";

cin >> ws;

getline(cin, name);

cout << "Enter level: ";

cin >> level;

}virtual void putdata() const {

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

cout << "Level: " << level << endl;

}};

class Warrior : public Character {

private:

int strength;

int defense;

public:

void getdata() override {

Character::getdata();

cout << "Enter strength: ";

cin >> strength;

cout << "Enter defense: ";

cin >> defense; }

void putdata() const override {

Character::putdata();

cout << "Strength: " << strength << endl;

cout << "Defense: " << defense << endl;}};

class Wizard : public Character {

private:

int mana;

int spellPower;

public:

void getdata() override {

Character::getdata();

cout << "Enter mana: ";

cin >> mana;

cout << "Enter spell power: ";

cin >> spellPower; }

void putdata() const override {

Character::putdata();

cout << "Mana: " << mana << endl;

cout << "Spell Power: " << spellPower << endl; }};

class Rogue : public Character {

private:

int agility;

int stealth;

int lockPickingSkill;

public:

void getdata() override {

Character::getdata();

cout << "Enter agility: ";

cin >> agility;

cout << "Enter stealth: ";

cin >> stealth;

cout << "Enter lock-picking skill: ";

cin >> lockPickingSkill; }

void putdata() const override {

Character::putdata();

cout << "Agility: " << agility << endl;

cout << "Stealth: " << stealth << endl;

cout << "Lock-picking Skill: " << lockPickingSkill << endl;}};

int main() {

Warrior warrior1;

Wizard wizard1;

Rogue rogue1;

cout << "Enter details for warrior 1:" << endl;

warrior1.getdata();

cout << endl;

cout << "Enter details for wizard 1:" << endl;

wizard1.getdata();

cout << endl;

cout << "Enter details for rogue 1:" << endl;

rogue1.getdata();

cout << endl;

cout << "Details of warrior 1:" << endl;

warrior1.putdata();

cout << endl;

cout << "Details of wizard 1:" << endl;

wizard1.putdata();

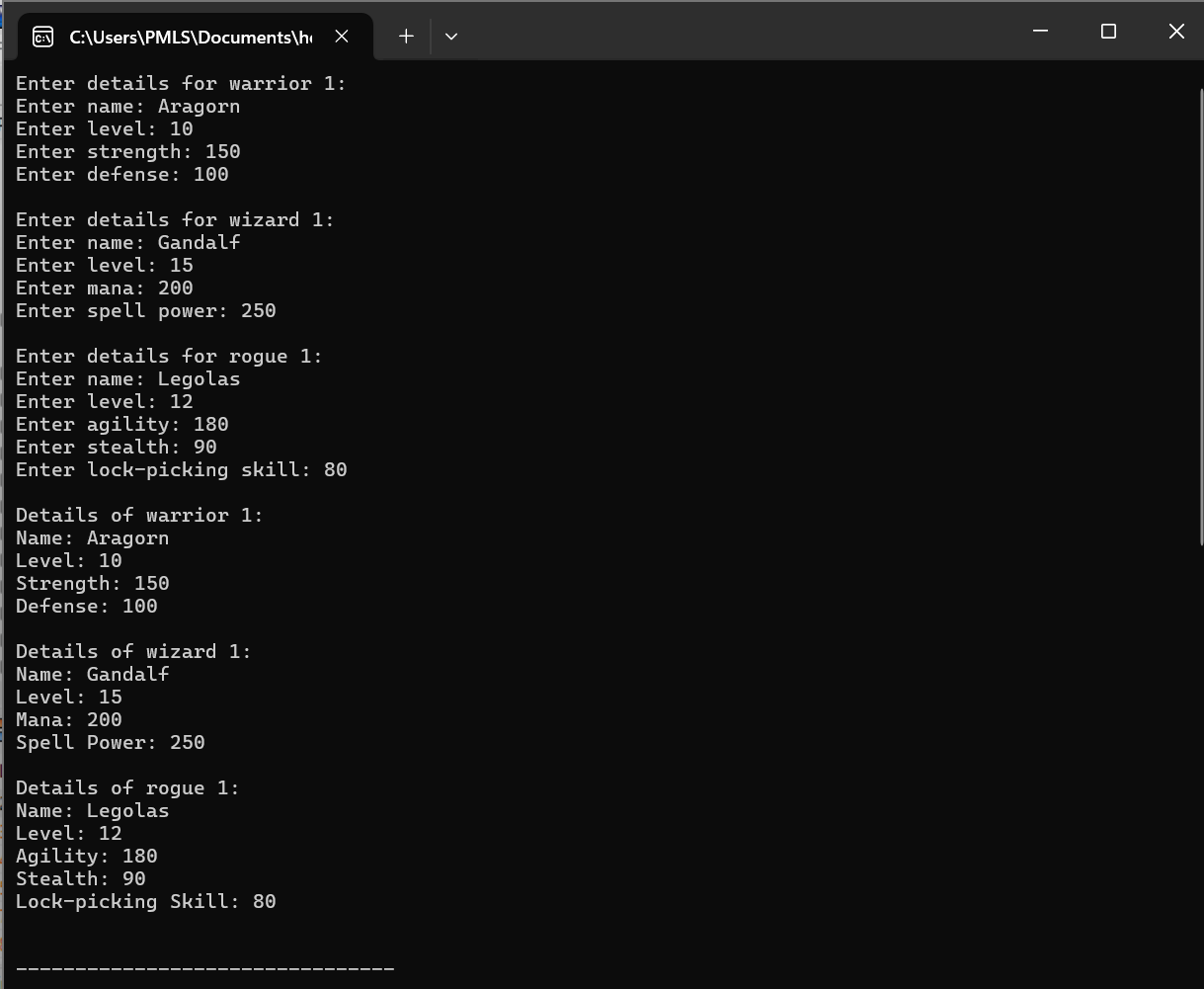
cout << endl;

cout << "Details of rogue 1:" << endl;

rogue1.putdata();

cout << endl;

return 0;}

**OUTPUT**

**Multilevel inheritance:**

You are tasked with implementing a C++ program that manages student information using multiple inheritance. You need to define the following classes:Class A:Attribute: rollNumber (integer) to store the student's roll number. Class B (derived from Class A):Attributes: subject1 and subject2 (both integers) to store marks for two subjects. Class C: (derived from Class B): Holds the total marks obtained by the student.Contains methods to calculate total marks and display all the details (roll number, marks of subjects, and total marks) Write the class definitions and a main() function that:Creates an instance of Class C. Displays all the details of the student.

**CODE:**

#include <iostream>

using namespace std;

class A{

int roll\_no;

public:

void getrollno(){

cout<<"Enter the Roll no :";

cin>>roll\_no;

}

void putrollno(){

cout<<"The Roll number is:"<<roll\_no<<endl;}

};

class B: public A{

protected:

int sub1,sub2;

public:

void getmarks(){

cout<<"Enter the Marks of sub 1 no :";

cin>>sub1;

cout<<"Enter the Marks of sub 2 no :";

cin>>sub2;

}

void putmarks(){

cout<<"The Marks of sub 1 is:"<<sub1<<endl;

cout<<"The Marks of sub 2 is:"<<sub2<<endl; }};

;

class C:public B{

public:

void total(){

putrollno();

putmarks();

cout<<"The total marks is :"<<sub1+sub2<<endl;

}};

int main(){

C obj;

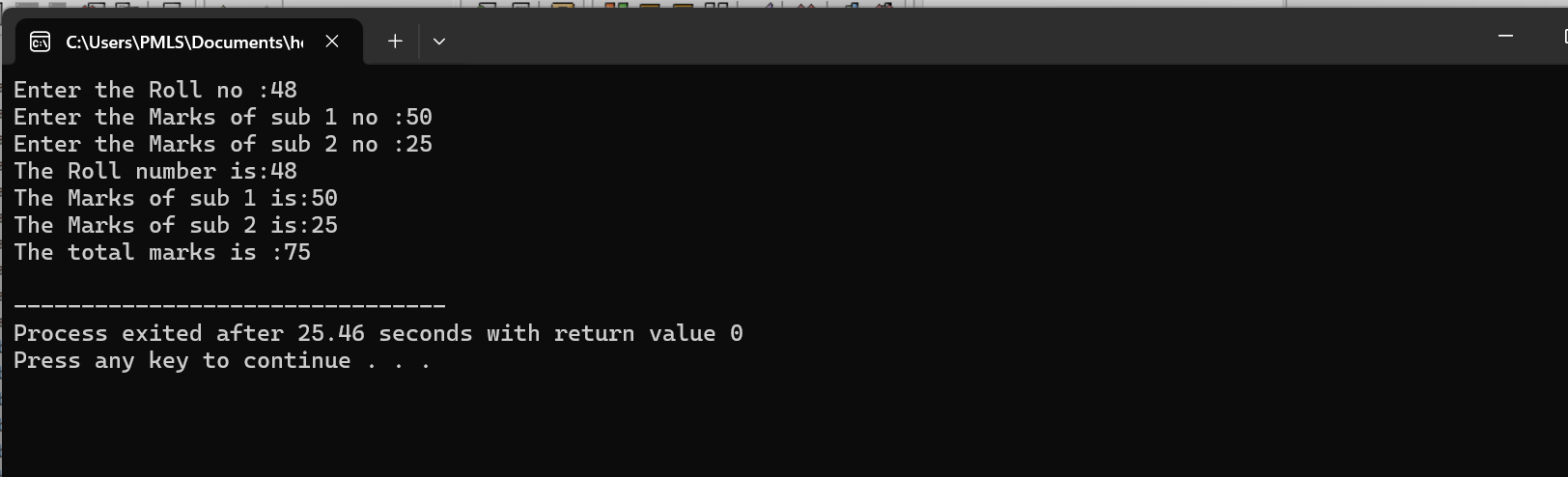
obj.getrollno();

obj.getmarks();

obj.total();

return 0;}

**OUTPUT**



**Multiple inheritance:**

Consider the following scenario involving multiple inheritance in C++:

Define a base class A with a protected member variable a.

Define another base class B with a protected member variable b.

Define a derived class C that inherits from both A and B. This class should have a member function sumVariables() that returns the sum of a and b.

**CODE:**

#include <iostream>

using namespace std;

class A {

protected:

int a;

public:

void input() {

cout << "Enter the value of A: ";

cin >> a;

}

};

class B {

protected:

int b;

public:

void getdata() {

cout << "Enter the value of B: ";

cin >> b;

}

};

class C : public A, public B {

public:

void sum\_variables() {

cout << "The Sum of variables is: " << a + b << endl;

}

};

int main() {

C obj;

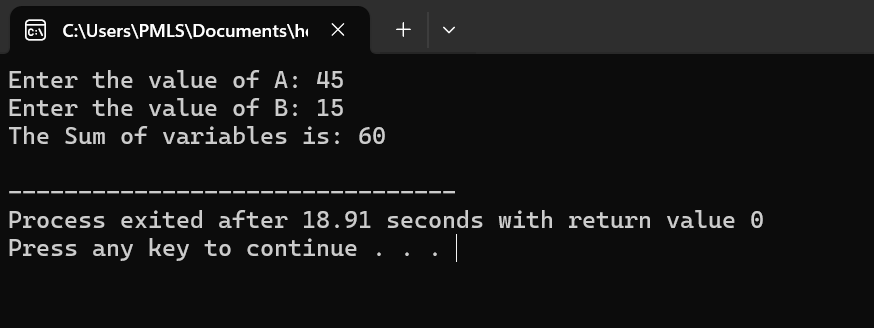
obj.input();

obj.getdata();

obj.sum\_variables();

return 0;

}

**OUTPUT:**