REHAN MUNIR

BCS221008

ASSIGNMENT#03

# Task 1: File Handling

* Create a class Student with data members: name, roll\_number, and marks.
* Implement member functions to input and display student data.
* Create a program to:
  + Create a file named students.txt.
  + Write data of 5 students to the file using ofstream.
  + Read the data from the file and display it using ifstream. o Modify the marks of a specific student using fstream.

CODE:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class Student {

public:

string name;

int roll\_number;

float marks;

// Function to input student data

void input() {

cout << "Enter name: ";

cin >> name;

cout << "Enter roll number: ";

cin >> roll\_number;

cout << "Enter marks: ";

cin >> marks;

}

// Function to display student data

void display() const {

cout << "Name: " << name << ", Roll Number: " << roll\_number << ", Marks: " << marks << endl;

}

// Function to write student data to file

void writeToFile(ofstream &out) const {

out << name << " " << roll\_number << " " << marks << endl;

}

// Function to read student data from file

void readFromFile(ifstream &in) {

in >> name >> roll\_number >> marks;

}

};

int main() {

const int studentCount = 5;

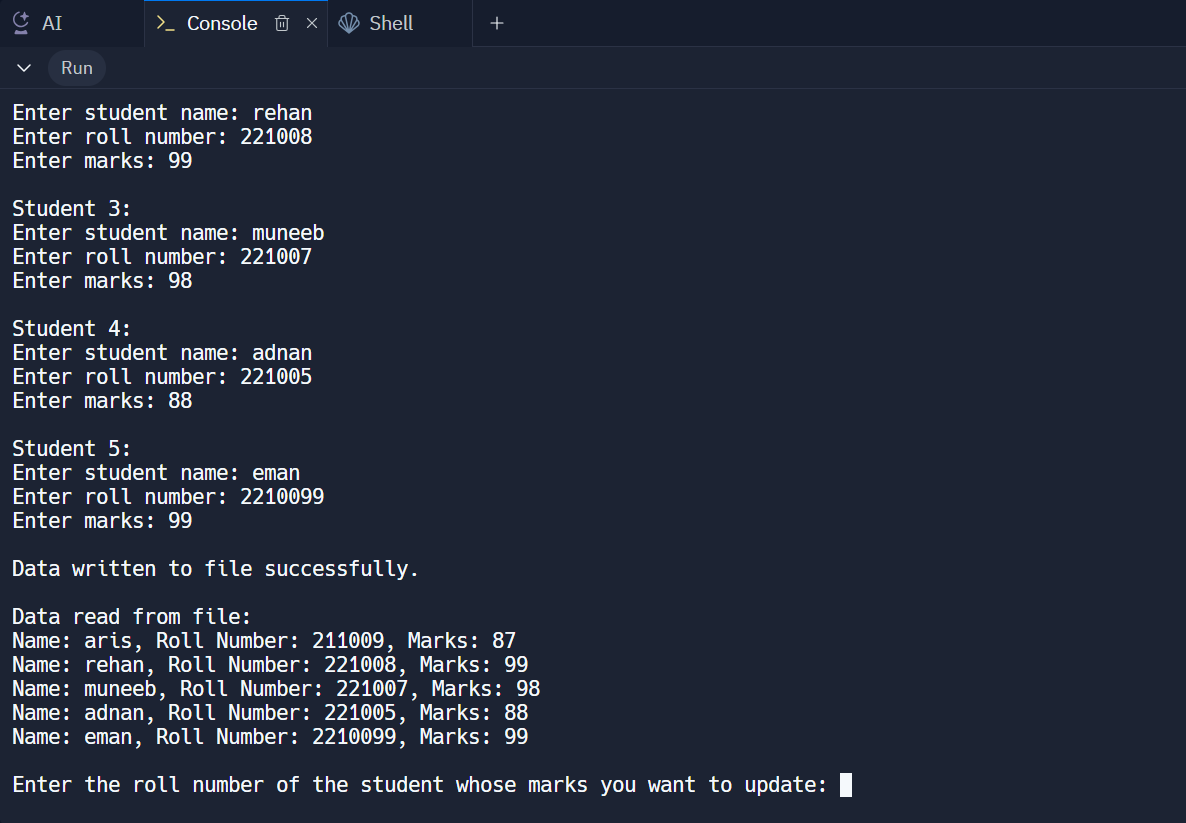
Student students[studentCount];

// Input student data

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

cout << "Enter details

OUTPUT:



**Task 2: Inheritance**

# Part A: Single Inheritance

* Create a base class Shape with data member color and member functions to set and get color.
* Create a derived class Rectangle inheriting Shape with data members length and breadth.
* Implement member functions to calculate area and perimeter of the rectangle.
* Demonstrate constructor chaining in both base and derived classes.

CODE:

#include <iostream>

#include <string>

using namespace std;

// Base class Shape

class Shape {

protected:

string color;

public:

// Constructor for Shape class

Shape(string col) : color(col) {

cout << "Creating a shape with color: " << color << endl;

}

// Function to set color

void setColor(string col) {

color = col;

}

// Function to get color

string getColor() const {

return color;

}

};

// Derived class Rectangle inheriting from Shape

class Rectangle : public Shape {

private:

float length;

float breadth;

public:

// Constructor for Rectangle class

Rectangle(string col, float len, float brd) : Shape(col), length(len), breadth(brd) {

cout << "Creating a rectangle with length: " << length << " and breadth: " << breadth << endl;

}

// Function to calculate area

float calculateArea() const {

return length \* breadth;

}

// Function to calculate perimeter

float calculatePerimeter() const {

return 2 \* (length + breadth);

}

// Function to display rectangle details

void displayDetails() const {

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

cout << "Color: " << getColor() << endl;

cout << "Length: " << length << ", Breadth: " << breadth << endl;

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

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

}

};

int main() {

string color;

float length, breadth;

// Getting user input

cout << "Enter the color of the rectangle: ";

cin >> color;

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

cin >> length;

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

cin >> breadth;

// Creating an object of Rectangle and demonstrating constructor chaining

Rectangle rect(color, length, breadth);

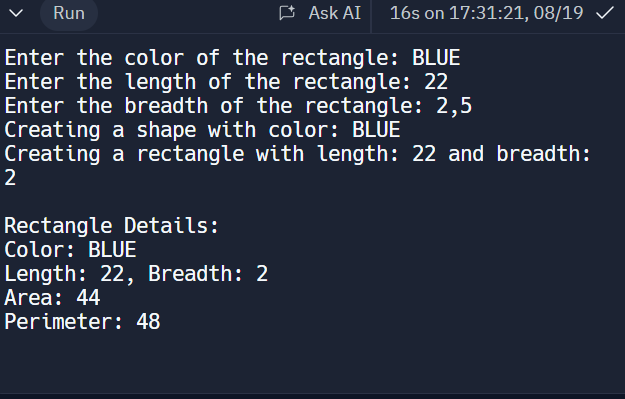
// Display rectangle details

rect.displayDetails();

return 0;

}

OUTPUT:



# Part B: Multilevel Inheritance

* Create a base class Animal with data member name.
* Create a derived class Mammal inheriting Animal with data member numberOfLegs.
* Create a derived class Dog inheriting Mammal with member functions to bark.
* Demonstrate object creation and access to members of all classes.

CODE:

#include <iostream>

#include <string>

using namespace std;

// Base class Animal

class Animal {

public:

string name;

Animal(string n) : name(n) {}

};

// Derived class Mammal inheriting Animal

class Mammal : public Animal {

public:

int numberOfLegs;

Mammal(string n, int legs) : Animal(n), numberOfLegs(legs) {}

};

// Derived class Dog inheriting Mammal

class Dog : public Mammal {

public:

Dog(string n, int legs) : Mammal(n, legs) {}

void bark() {

cout << name << " is barking: Woof! Woof!" << endl;

}

};

int main() {

// Creating a Dog object

Dog myDog("Buddy", 4);

// Accessing and displaying details

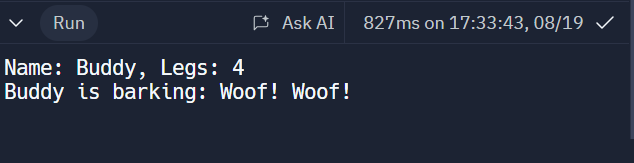
cout << "Name: " << myDog.name << ", Legs: " << myDog.numberOfLegs << endl;

myDog.bark();

return 0;

}

OUTPUT:



# Part C: Multiple Inheritance

* Create a base class Person with data members name and address.
* Create another base class Employee with data members employeeID and salary.
* Create a derived class Teacher inheriting both Person and Employee with additional data member subject.
* Demonstrate object creation and access to members of all classes.
* Discuss the ambiguity problem in multiple inheritance and how to resolve it.

CODE

#include <iostream>

#include <string>

using namespace std;

// Base class Person

class Person {

public:

string name;

string address;

Person(string n, string addr) : name(n), address(addr) {}

};

// Base class Employee

class Employee {

public:

int employeeID;

double salary;

Employee(int id, double sal) : employeeID(id), salary(sal) {}

};

// Derived class Teacher inheriting from both Person and Employee

class Teacher : public Person, public Employee {

public:

string subject;

Teacher(string n, string addr, int id, double sal, string subj)

: Person(n, addr), Employee(id, sal), subject(subj) {}

void displayDetails() {

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

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

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

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

cout << "Subject: " << subject << endl;

}

};

int main() {

// Creating a Teacher object

Teacher myTeacher("John Doe", "123 Main St", 1001, 50000, "Mathematics");

// Displaying details

myTeacher.displayDetails();

return 0;

}

OUTPUT:

