//Basic Concepts of OOP

//1. WAP to print “Hello World” using C++

#include<iostream>

using namespace std;

int main()

{

cout<<"Hello World";

}

//2. What is OOP? List OOP concepts

Ans : In c++ programming language this is most important thing to understand

OOP is known as “Object Oriented Programming System”.

OOP have 5 concept for high level programming.

Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

OOPS Concept:

1.Class & Object

2.Inheritance

3.Polymorphism

4.Encapsulation

5.Abstracntion

//3. What is the difference between OOP and POP?

Ans **:**

**(1).OOPS And POP Difference**

|  |  |
| --- | --- |
| **Program is divided into objects.** | **Program is divided into functions.** |
| **Bottom-up approach.** | **Top-down approach.** |
| **Inheritance property is used.** | **Inheritance is not allowed.** |
| **It uses access specifier.** | **It doesn’t use access specifier.** |
| **Encapsulation is used to hide the data.** | **No data hiding.** |
| **Concept of virtual function.** | **No virtual function.** |
| **Object functions are linked through message passing.** | **Parts of program are linked through parameter passing.** |
| **Adding new data and functions is easy** | **Expanding new data and functions is not easy.** |
| **The existing code can be reused.** | **No code reusability.** |
| **use for solving big problems.** | **Not suitable for solving big problems.** |
| **C++ , Java** | **C , Pascal** |

1. WAP to create simple calculator using class

Ans:

//1. WAP to create simple calculator using class

#include<iostream>

using namespace std;

class Calculator

{

public:

myfun1()

{

int a,b;

cout<<"Enter number 1:";

cin>>a;

cout<<"Enter number 2:";

cin>>b;

cout<<"Addition of numbers:"<<a+b<<endl;

cout<<"Subtraction of numbers:"<<a-b<<endl;

cout<<"Multiplication of numbers:"<<a\*b<<endl;

cout<<"Dividor of numbers:"<<a/b;

}

};

int main()

{

Calculator obj;

obj.myfun1();

}

//2. Define a class to represent a bank account. Include the following members:

#include<iostream>

using namespace std;

class Bank

{

public:

int age,mobile\_no,pincode,ac\_no=2207,balance=20000;

string name,email\_id,address;

int depo,withdr,checkbal;

//method 1st about account create

ac\_create()

{

cout<<"Enter user fullname:";

cin>>name;

cout<<"Enter email id:";

cin>>email\_id;

cout<<"Enter age of user:";

cin>>age;

cout<<"Enter mobile number of user:";

cin>>mobile\_no;

cout<<"Enter pincode number:";

cin>>pincode;

cout<<"Your Account Number is:"<<ac\_no<<endl;

cout<<"Your Deposit Balance is:"<<balance<<endl;

this->balance=balance;

}

//method 2 of deposit balance

deposit()

{

cout<<"Enter the amount of deposit:";

cin>>depo;

this->balance +=depo;

cout<<"Deposit amount is:"<<depo<<endl;

}

//Method 3 of withrwal amount

withdrwal()

{

cout<<"Enter amount of Withdrwal:";

cin>>withdr;

this->balance -=withdr;

cout<<"Withdrwal amount is:"<<withdr<<endl;

if(withdr>balance)

{

cout<<"Not enough balance"<<endl;

}

else

{

cout<<"run the process"<<endl;

}

}

//method 4 of check balance

check\_balance()

{

cout<<"Your check balance is:"<<this->balance;

}

};

int main()

{

Bank obj;

cout<<"Press 1 for create a account:"<<endl;

cout<<"press 2 for exit"<<endl;

int ch;

cout<<"Enter choice:";

cin>>ch;

if(ch==1)

{

obj.ac\_create();

while(1)

{

cout<<"Press 1 for depoiste"<<endl;

cout<<"Press 2 for withdrawl"<<endl;

cout<<"Press 3 for check balance"<<endl;

cout<<"Invalid choice"<<endl;

int ch1;

cout<<"Enter choice:"<<endl;

cin>>ch1;

if(ch1==1)

{

obj.deposit();

}

else if(ch1==2)

{

obj.withdrwal();

}

else if(ch1==3)

{

obj.check\_balance();

}

else if(ch1==4)

{

cout<<"Thank you";

break;

}

else

{

cout<<"Invalid choice";

break;

}

}

}

}

//3.Data Member: -Name of the depositor -Account Number -Type of Account -Balance amount in the account Member Functions -To assign values -To deposited an amount -To withdraw an amount after checking balance -To display name and balance

#include<iostream>

using namespace std;

//Bank management system

class Bank{

public:

string name,email\_id;

int age,ac\_no=2209,balance=10000;

int de,we,cbal;

//method 1 user account details

ac\_create()

{

cout<<"Enter user name:";

cin>>name;

cout<<"Enter email id:";

cin>>email\_id;

cout<<"Enter user age:";

cin>>age;

cout<<"your account number is:"<<ac\_no<<endl;

cout<<"your deposit balance is:"<<balance<<endl;

this->balance=balance;

}

//method 2 deposite balance

deposite()

{

cout<<"Enter amount for deposite is:";

cin>>de;

this->balance +=de;

cout<<"Deposite amount is:"<<de<<endl;

}

//method 3 withdrwals amount

withdrwal()

{

cout<<"Enter withdrwal amount:";

cin>>we;

this-> balance-=we;

cout<<"Withdrwal amount is:"<<we<<endl;

if(we>balance)

{

cout<<"Not enough balance"<<endl;

}

else

{

cout<<"run the process"<<endl;

}

}

//method 3 after the withdrawl the perfect amount is

check\_balance()

{

cout<<"Check balannce is:"<<this->balance;

}

};

int main()

{

Bank obj;

cout<<"Press 1 for create a account:"<<endl;

cout<<"press 2 for exit"<<endl;

//for condtions if user create an account

int ch;

cout<<"Enter choice :";

cin>>ch;

if(ch==1)

{

obj.ac\_create();

while (1)

{

cout<<"Press 1 for deposite"<<endl;

cout<<"press 2 for withdrawl"<<endl;

cout<<"press 3 for check balance"<<endl;

cout<<"Invalid choice"<<endl;

int ch1;

cout<<"Enter choice:"<<endl;

cin>>ch1;

if(ch1==1)

{

obj.deposite();

}

else if(ch1==2)

{

obj.withdrwal();

}

else if(ch1==3)

{

obj.check\_balance();

}

else if(ch1==4)

{

cout<<"thank you";

break;

}

else

{

cout<<"Invalid choice";

break;

}

}

}

return 0;

}

//4. Write a C++ program to implement a class called Circle that has private member variables for radius. Include member functions to calculate the circle's area and circumference. in c++

#include <iostream>

#include <cmath>

const double PI = 3.14159; // Define the value of PI as a constant

class Circle { // Define a class named Circle

private:

double radius; // Private member to store the radius

public:

// Constructor to initialize the Circle object with a radius

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

// Member function to calculate the area of the circle

double calculateArea() {

return PI \* pow(radius, 2); // Formula to calculate the area of a circle

}

// Member function to calculate the circumference of the circle

double calculateCircumference() {

return 2 \* PI \* radius; // Formula to calculate the circumference of a circle

}

};

int main() {

// Create a circle object

double radius;

std::cout << "Input the radius of the circle: ";

std::cin >> radius; // Input the radius from the user

Circle circle(radius); // Create a Circle object with the given radius

// Calculate and display the area of the circle

double area = circle.calculateArea(); // Calculate the area using the Circle object

std::cout << "Area: " << area << std::endl; // Output the calculated area

// Calculate and display the circumference of the circle

double circumference = circle.calculateCircumference(); // Calculate the circumference using the Circle object

std::cout << "Circumference: " << circumference << std::endl; // Output the calculated circumference

return 0; // Return 0 to indicate successful completion

}

//5. Write a C++ program to create a class called Rectangle that has private member variables for length and width. Implement member functions to calculate the rectangle's area and perimeter.

//Sure, I'd be happy to help you with that! Below is a C++ program that defines a class called Rectangle with private member variables for length and width. It also includes member functions to calculate the area and perimeter of the rectangle.

#include <iostream>

class Rectangle {

private:

double length;

double width;

public:

// Constructor to initialize length and width

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

// Member function to calculate area

double calculateArea() const {

return length \* width;

}

// Member function to calculate perimeter

double calculatePerimeter() const {

return 2 \* (length + width);

}

// Member function to display the dimensions, area, and perimeter

void display() const {

std::cout << "Length: " << length << "\n";

std::cout << "Width: " << width << "\n";

std::cout << "Area: " << calculateArea() << "\n";

std::cout << "Perimeter: " << calculatePerimeter() << "\n";

}

};

int main() {

// Create a Rectangle object

Rectangle rect(10.0, 5.0);

// Display the rectangle's dimensions, area, and perimeter

rect.display();

return 0;

}

//6. Write a C++ program to create a class called Person that has private member variables for name, age and country. Implement member functions to set and get the values of these variables.

#include <iostream>

#include <string>

class Person {

private:

std::string name;

int age;

std::string country;

public:

// Setter functions to set the private member variables

void setName(const std::string & n) { // Setter function for the name

name = n; // Assign the provided name to the 'name' member variable

}

void setAge(int a) { // Setter function for the age

age = a; // Assign the provided age to the 'age' member variable

}

void setCountry(const std::string & c) { // Setter function for the country

country = c; // Assign the provided country to the 'country' member variable

}

// Getter functions to retrieve the private member variables

std::string getName() { // Getter function for retrieving the name

return name; // Return the stored name

}

int getAge() { // Getter function for retrieving the age

return age; // Return the stored age

}

std::string getCountry() { // Getter function for retrieving the country

return country; // Return the stored country

}

};

int main() {

// Create a person object

Person person;

// Set the person's details using setter functions

person.setName("Vinay patel"); // Set the person's name

person.setAge(20); // Set the person's age

person.setCountry("INDIA"); // Set the person's country

// Get and display the person's details using getter functions

std::cout << "Name: " << person.getName() << std::endl; // Output the person's name

std::cout << "Age: " << person.getAge() << std::endl; // Output the person's age

std::cout << "Country: " << person.getCountry() << std::endl; // Output the person's country

return 0; // Return 0 to indicate successful completion

}

Constructor Destructor Encapsulation Abstraction

//1. Write a program to find the multiplication values and the cubic values using inline function

Sure! Here's a C++ program that uses inline functions to find the multiplication values and the cubic values of given numbers:

#include <iostream>

// Inline function to calculate multiplication

inline int multiply(int a, int b) {

return a \* b;

}

// Inline function to calculate cubic value

inline int cubic(int x) {

return x \* x \* x;

}

int main() {

int num1, num2;

std::cout << "Enter two numbers to multiply: ";

std::cin >> num1 >> num2;

std::cout << "Multiplication of " << num1 << " and " << num2 << " is: " << multiply(num1, num2) << std::endl;

int num;

std::cout << "Enter a number to find its cubic value: ";

std::cin >> num;

std::cout << "Cubic value of " << num << " is: " << cubic(num) << std::endl;

return 0;

}

//2. Write a program of Addition, Subtraction, Division, Multiplication using constructor.

#include<iostream>

using namespace std;

class A

{

public:

myfun1(int a,int b)

{

cout<<"Addition of the numbers:"<<a+b<<endl;

cout<<"Subtraction of the number:"<<a-b<<endl;

cout<<"Multiplication of the number:"<<a\*b<<endl;

cout<<"Divison of the number:"<<a/b<<endl;

}

};

int main()

{

A obj;

obj.myfun1(10,2);

}

//3. Write a C++ program to create a class called Car that has private member variables for company, model, and year. Implement member functions to get and set these variables.

#include <iostream>

#include <string>

class Car {

private:

std::string company; // Private member to store the company name of the car

std::string model; // Private member to store the model name of the car

int year; // Private member to store the year of the car

public:

// Constructor to initialize Car object with provided values

Car(const std::string & comp, const std::string & mdl, int yr): company(comp), model(mdl), year(yr) {}

// Getter functions to retrieve private member variables

std::string getCompany() { // Getter function for retrieving the company name

return company; // Return the stored company name

}

std::string getModel() { // Getter function for retrieving the model name

return model; // Return the stored model name

}

int getYear() { // Getter function for retrieving the year

return year; // Return the stored year

}

// Setter functions to modify private member variables

void setCompany(const std::string & comp) { // Setter function for modifying the company name

company = comp; // Assign the provided company name to the 'company' member variable

}

void setModel(const std::string & mdl) { // Setter function for modifying the model name

model = mdl; // Assign the provided model name to the 'model' member variable

}

void setYear(int yr) { // Setter function for modifying the year

year = yr; // Assign the provided year to the 'year' member variable

}

};

int main() {

// Create a car object with initial values

Car car("AUDI", "A6", 2023);

// Get and display the car details using getter functions

std::cout << "Company: " << car.getCompany() << std::endl; // Output the car's company name

std::cout << "Model: " << car.getModel() << std::endl; // Output the car's model name

std::cout << "Year: " << car.getYear() << std::endl; // Output the car's year

// Set new values for the car using setter functions

car.setCompany("BMW"); // Update the car's company name

car.setModel("M4"); // Update the car's model name

car.setYear(2022); // Update the car's year

// Get and display the updated car details using getter functions

std::cout << "\nUpdated Company: " << car.getCompany() << std::endl; // Output the updated car's company name

std::cout << "Updated Model: " << car.getModel() << std::endl; // Output the updated car's model name

std::cout << "Updated Year: " << car.getYear() << std::endl; // Output the updated car's year

return 0; // Return 0 to indicate successful completion

}

//4. Write a C++ program to implement a class called Bank Account that has private member variables for account number and balance. Include member functions to deposit and withdraw money from the account.

#include <iostream>

using namespace std;

class BankAccount {

private:

int accountNumber;

double balance;

public:

// Constructor to initialize account number and balance

BankAccount(int accNum, double initialBalance) {

accountNumber = accNum;

balance = initialBalance;

}

// Function to deposit money into the account

void deposit(double amount) {

if (amount > 0) {

balance += amount;

cout << "Deposited: " << amount << ". New balance: " << balance << endl;

} else {

cout << "Invalid deposit amount." << endl;

}

}

// Function to withdraw money from the account

void withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

cout << "Withdrew: " << amount << ". New balance: " << balance << endl;

} else {

cout << "Invalid withdrawal amount or insufficient balance." << endl;

}

}

// Function to display account details

void display() const {

cout << "Account Number: " << accountNumber << ", Balance: " << balance << endl;

}

};

int main()

{

// Create a BankAccount object

BankAccount myAccount(123456, 1000.0);

// Display initial account details

myAccount.display();

// Deposit money

myAccount.deposit(500.0);

// Withdraw money

myAccount.withdraw(200.0);

// Attempt to withdraw more money than available

myAccount.withdraw(1500.0);

// Display final account details

myAccount.display();

return 0;

}

//5. Write a C++ program to create a class called Triangle that has private member variables for the lengths of its three sides. Implement member functions to determine if the triangle is equilateral, isosceles, or scalene.

#include <iostream>

using namespace std;

class Triangle {

private:

double side1, side2, side3;

public:

// Constructor to initialize the sides of the triangle

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

// Function to check if the triangle is equilateral

bool isEquilateral() {

return (side1 == side2 && side2 == side3);

}

// Function to check if the triangle is isosceles

bool isIsosceles() {

return (side1 == side2 || side2 == side3 || side1 == side3);

}

// Function to check if the triangle is scalene

bool isScalene() {

return (side1 != side2 && side2 != side3 && side1 != side3);

}

// Function to display the type of triangle

void displayType() {

if (isEquilateral()) {

cout << "The triangle is equilateral." << endl;

} else if (isIsosceles()) {

cout << "The triangle is isosceles." << endl;

} else if (isScalene()) {

cout << "The triangle is scalene." << endl;

} else {

cout << "Invalid triangle." << endl;

}

}

};

int main()

{

Triangle t1(3.0, 3.0, 3.0);

t1.displayType(); // Output: The triangle is equilateral.

Triangle t2(3.0, 4.0, 4.0);

t2.displayType(); // Output: The triangle is isosceles.

Triangle t3(3.0, 4.0, 5.0);

t3.displayType(); // Output: The triangle is scalene.

return 0;

}

//6. Write a C++ program to implement a class called Employee that has private member variables for name, employee ID, and salary. Include member functions to calculate and set salary based on employee performance. Using of constructor

#include <iostream>

#include <string>

class Employee {

private:

std::string name;

int employeeID;

double salary;

public:

// Constructor to initialize the employee object

Employee(std::string empName, int empID, double empSalary)

: name(empName), employeeID(empID), salary(empSalary) {}

// Function to set the salary based on performance

void setSalary(double performanceRating) {

if (performanceRating >= 4.5) {

salary \*= 1.20; // 20% increase for excellent performance

} else if (performanceRating >= 3.5) {

salary \*= 1.10; // 10% increase for good performance

} else if (performanceRating >= 2.5) {

salary \*= 1.05; // 5% increase for average performance

} else {

salary \*= 0.90; // 10% decrease for poor performance

}

}

// Function to display employee details

void display() const {

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

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

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

}

};

int main()

{

// Creating an Employee object

Employee emp1("Vinay Patel", 12345, 50000.0);

// Display initial details

emp1.display();

// Set salary based on performance rating

emp1.setSalary(4.7); // Example performance rating

// Display updated details

emp1.display();

return 0;

}

//7. Write a C++ program to implement a class called Date that has private member variables for day, month, and year. Include member functions to set and get these variables, as well as to validate if the date is valid.

#include <iostream>

class Date {

private:

int day;

int month;

int year;

bool isLeapYear(int year) {

return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);

}

bool isValidDate(int day, int month, int year) {

if (year < 1 || month < 1 || month > 12 || day < 1) {

return false;

}

int daysInMonth[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

if (isLeapYear(year) && month == 2) {

daysInMonth[1] = 29;

}

return day <= daysInMonth[month - 1];

}

public:

void setDate(int d, int m, int y) {

if (isValidDate(d, m, y)) {

day = d;

month = m;

year = y;

} else {

std::cerr << "Invalid date!" << std::endl;

}

}

int getDay() const {

return day;

}

int getMonth() const {

return month;

}

int getYear() const {

return year;

}

void displayDate() const {

std::cout << day << "/" << month << "/" << year << std::endl;

}

};

int main() {

Date date;

date.setDate(29, 2, 2024); // Example of a valid leap year date

date.displayDate();

date.setDate(31, 4, 2023); // Example of an invalid date

date.displayDate();

return 0;

}

//8. Write a C++ program to implement a class called Student that has private member variables for name, class, roll number, and marks. Include member functions to calculate the grade based on the marks and display the student's information. Accept address from each student implementusing of aggregation

#include <iostream> // Include necessary header for input/output stream

#include <string> // Include necessary header for string handling

class Student { // Define a class named Student

private:

std::string name; // Private member variable to store student's name

std::string studentClass; // Private member variable to store student's class

int rollNumber; // Private member variable to store student's roll number

double marks; // Private member variable to store student's marks

public:

// Constructor to initialize Student object with provided details

Student(const std::string & studentName, const std::string & sClass, int rollNum, double studentMarks)

: name(studentName), studentClass(sClass), rollNumber(rollNum), marks(studentMarks) {}

// Member function to calculate the grade based on marks

std::string calculateGrade() {

if (marks >= 90) {

return "A+";

} else if (marks >= 80) {

return "A";

} else if (marks >= 70) {

return "B";

} else if (marks >= 60) {

return "C";

} else {

return "D";

}

}

// Member function to display student information

void displayInformation() {

std::cout << "\n\nName: " << name << std::endl; // Output student's name

std::cout << "Class: " << studentClass << std::endl; // Output student's class

std::cout << "Roll Number: " << rollNumber << std::endl; // Output student's roll number

std::cout << "Marks: " << marks << std::endl; // Output student's marks

std::cout << "Grade: " << calculateGrade() << std::endl; // Output student's grade

}

};

int main() {

// Create a student object

std::string studentName; // Variable to store student's name

std::string sClass; // Variable to store student's class

int rollNum; // Variable to store student's roll number

double studentMarks; // Variable to store student's marks

std::cout << "Student details: "; // Prompt user for student details

std::cout << "\nName: "; // Prompt user for student's name

std::getline(std::cin, studentName); // Input student's name

std::cout << "Class: "; // Prompt user for student's class

std::getline(std::cin, sClass); // Input student's class

std::cout << "Roll number: "; // Prompt user for student's roll number

std::cin >> rollNum; // Input student's roll number

std::cout << "Marks (0-100): "; // Prompt user for student's marks

std::cin >> studentMarks; // Input student's marks

Student student(studentName, sClass, rollNum, studentMarks); // Create Student object with provided details

// Display student information

student.displayInformation();

return 0;

}

Inheritance Polymorphism

//1. Assume a class cricketer is declared. Declare a derived class batsman from cricketer. Data member of batsman. Total runs, Average runs and best performance. Member functions input data, calculate average runs, Display data. (Single Inheritance)

#include<iostream>

using namespace std;

class Cricketer

{

string name;

public:

batsman()

{

cout<<"Enter the Batsman Name:";

cin>>name;

}

};

class B:public Cricketer{

int runs,avgruns;

int calavg\_runs;

public:

myfun1()

{

cout<<"Total Runs=";

cin>>runs;

avgruns=runs/3;

cout<<"Average runs of batsman is:"<<avgruns<<endl;

cout<<"Best Performance is in last match 121"<<endl;

calavg\_runs=runs+avgruns;

cout<<" Best Average runs is ="<<calavg\_runs;

}

};

int main()

{

B obj;

obj.batsman();

obj.myfun1();

}

//2. Write a C++ Program to find Area of Rectangle using inheritance

#include<iostream>

using namespace std;

class A{

public:

myfun1()

{

int w,l; // Area of Rectangle a=wl

cout<<"Enter the widths:";

cin>>w;

cout<<"Enter the length:";

cin>>l;

}

};

class B:public A

{

public:

myfun2()

{

int a,w,l;

a=w\*l;

cout<<"Area of Rectancle is:"<<a;

}

};

int main()

{

B obj;

obj.myfun1();

obj.myfun2();

}

//3. Create a class person having members name and age. Derive a class student having member percentage. Derive another class teacher having member salary. Write necessary member function to initialize, read and write data. Write also Main function (Multiple Inheritance)

#include<iostream>

using namespace std;

class Person

{

public:

myfun1(int age,string name)

{

cout<<"Enter name:"<<name<<endl;

cout<<"Enter age:"<<age<<endl;

}

};

class Student

{

public:

myfun2(float percentage)

{

cout<<"Student percentage is:"<<percentage<<endl;

}

};

class Teacher:public Person,public Student

{

public:

myfun3(int salary)

{

cout<<"Teacher's salary is:"<<salary<<endl;

}

};

int main()

{

Teacher obj;

obj.myfun1(21,"Vinay Patel");

obj.myfun2(75.09);

obj.myfun3(25000);

}

//4. Write a C++ Program display Student Mark sheet using Multiple inheritance

#include<iostream>

using namespace std;

class Student

{

public:

myfun1()

{

int enroll\_no;

string name,branch;

cout<<"Enter Student Name:";

cin>>name;

cout<<"Enter Student's Enrollment no:";

cin>>enroll\_no;

cout<<"Branch :";

cin>>branch;

}

};

class Subjects

{

public:

myfun2()

{

int Py,Ada,Pe,Cn,Se;

cout<<"Enter marks of Python out of 100:";

cin>>Py;

cout<<"Enter marks of Analysis Of Algorithms out of 100:";

cin>>Ada;

cout<<"Enter marks of Professional Ethics out of 100:";

cin>>Pe;

cout<<"Enter marks of Computer Network out of 100:";

cin>>Cn;

cout<<"Enter marks of Software Engineering out of 100:";

cin>>Se;

}

};

class Percentage:public Student,public Subjects

{

public:

myfun3()

{

int Py,Ada,Pe,Cn,Se;

int total;

float percentage;

total= Py+Ada+Pe+Cn+Se;

cout<<"Total marks is:"<<total<<endl;

percentage = total/5;

cout<<"Percentage :"<<percentage;

}

};

int main()

{

Percentage obj;

obj.myfun1();

obj.myfun2();

obj.myfun3();

}

//5. Assume that the test results of a batch of students are stored in three different classes. Class Students are storing the roll number. Class Test stores the marks obtained in two subjects and class result contains the total marks obtained in the test. The class result can inherit the details of the marks obtained in the test and roll number of students. (Multilevel Inheritance)

#include<iostream>

using namespace std;

class Student

{

public:

myfun1( int rollno)

{

cout<<"Roll no is:"<<rollno<<endl;

}

};

class Marks:public Student{

public:

myfun2()

{

int m1,m2;

cout<<"Enter the marks of subject 1:";

cin>>m1;

cout<<"Enter the marks of subject 2:";

cin>>m2;

}

};

class Result:public Marks

{

public:

myfun3()

{

int total,m1,m2;

total=m1+m2;

cout<<"Result out of 150 :"<<total<<endl;

}

};

int main()

{

Result obj;

obj.myfun1(21);

obj.myfun2();

obj.myfun3();

}

//6. Write a C++ Program to show access to Private Public and Protected using Inheritance

class Base {

public:

int x;

protected:

int y;

private:

int z;

};

class PublicDerived: public Base {

// x is public

// y is protected

// z is not accessible from PublicDerived

};

class ProtectedDerived: protected Base {

// x is protected

// y is protected

// z is not accessible from ProtectedDerived

};

class PrivateDerived: private Base {

// x is private

// y is private

// z is not accessible from PrivateDerived

};

int main()

{

}

//7. Write a C++ Program to illustrates the use of Constructors in multilevel inheritance

// C++ program to implement

// Multilevel Inheritance

#include <bits/stdc++.h>

using namespace std;

// single base class

class A {

public:

int a;

void get\_A\_data()

{

cout << "Enter value of a: ";

cin >> a;

}

};

// derived class from base class

class B : public A {

public:

int b;

void get\_B\_data()

{

cout << "Enter value of b: ";

cin >> b;

}

};

// derived from class derive1

class C : public B {

private:

int c;

public:

void get\_C\_data()

{

cout << "Enter value of c: ";

cin >> c;

}

// function to print sum

void sum()

{

int ans = a + b + c;

cout << "sum: " << ans;

}

};

int main()

{

// object of sub class

C obj;

obj.get\_A\_data();

obj.get\_B\_data();

obj.get\_C\_data();

obj.sum();

return 0;

}

//8. Write a program to Mathematic operation like Addition, Subtraction, Multiplication, Division Of two number using different parameters and Function Overloading

#include<iostream>

using namespace std;

class A

{

public:

myfun1()

{

int a=10,b=2;

cout<<"Additon of the numbers:"<<a+b<<endl;

}

myfun2(int n)

{

int a,b;

cout<<"Substraction of two numbers:"<<a-b<<endl;

}

myfun3( int n,int f)

{

int a,b;

cout<<"Multiplication of two numbers:"<<a\*b<<endl;

}

myfun4(int d,int g,int m)

{

int a,b;

cout<<"Divison of two numbers:"<<a/b<<endl;

}

};

int main()

{

A obj;

obj.myfun1();

obj.myfun2(12);

obj.myfun3(10,20);

obj.myfun4(21,22,23);

}

//9. Write a Program of Two 1D Matrix Addition using Operator Overloading

#include <iostream>

#include <vector>

class Matrix {

private:

std::vector<int> data;

public:

Matrix(int size) : data(size) {}

void input() {

for (int &element : data) {

std::cin >> element;

}

}

Matrix operator+(const Matrix &other) {

int size = data.size();

Matrix result(size);

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

result.data[i] = this->data[i] + other.data[i];

}

return result;

}

void display() const {

for (const int &element : data) {

std::cout << element << " ";

}

std::cout << std::endl;

}

};

int main()

{

int size;

std::cout << "Enter the size of the matrices: ";

std::cin >> size;

Matrix mat1(size), mat2(size);

std::cout << "Enter elements of first matrix: ";

mat1.input();

std::cout << "Enter elements of second matrix: ";

mat2.input();

Matrix result = mat1 + mat2;

std::cout << "Resultant matrix after addition: ";

result.display();

return 0;

}

//10.Write a program to concatenate the two strings using Operator Overloading

// C++ Program to concatenate two string

// using unary operator overloading

#include <iostream>

#include <string.h>

using namespace std;

// Class to implement operator overloading

// function for concatenating the strings

class AddString {

public:

// Classes object of string

char s1[25], s2[25];

// Parameterized Constructor

AddString(char str1[], char str2[])

{

// Initialize the string to class object

strcpy(this->s1, str1);

strcpy(this->s2, str2);

}

// Overload Operator+ to concat the string

void operator+()

{

cout << "\nConcatenation: " << strcat(s1, s2);

}

};

// Driver Code

int main()

{

// Declaring two strings

char str1[] = "Vinay";

char str2[] = "Patel";

// Declaring and initializing the class

// with above two strings

AddString a1(str1, str2);

// Call operator function

+a1;

return 0;

}

//11. Write a program to calculate the area of circle, rectangle and triangle using Function Overloading

//Rectangle: Area \* breadth

//Triangle: ½ \*Area\* breadth

//Circle: Pi \* Area \*Area

#include<iostream>

using namespace std;

class A

{

public:

rectangle()//20

{

int a,b;

cout<<"Enter the area:";

cin>>a;

cout<<"Enter the breadth:";

cin>>b;

cout<<"Area of Rectangle is:"<<a\*b<<endl;

}

triangle(int f)//10

{

int a,b;

cout<<"Area of Rectangle is:"<<0.5\*a\*b<<endl;

}

circle(int e, int g)//62.8

{

int a,b;

cout<<"Area of Rectangle is:"<<3.14\*a\*b<<endl;

}

};

int main()

{

A obj;

obj.rectangle();

obj.triangle(1);

obj.circle(1,1);

}

//12. Write a program to swap the two numbers using friend function without using third variable

#include<iostream>

using namespace std;

class A

{

private:

int a=10,b=20;

friend show(A&obj);

};

show(A&obj)

{

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

cout<<" value of b="<<obj.b<<endl;

int temp;

temp=obj.a;

obj.a=obj.b;

obj.b=temp;

cout<<" After the swapping value of a="<<obj.a<<endl;

cout<<" After the swapping value of b="<<obj.b;

}

int main()

{

A obj;

show(obj);

return 0;

}

//13. Write a program to find the max number from given two numbers using friend function

#include<iostream>

using namespace std;

class A

{

private:

int a,b;

friend show(A&obj);

};

show(A&obj)

{

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

cin>>obj.a;

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

cin>>obj.b;

if(obj.a== obj.b)

{

cout<<"Both have same value:"<<obj.a,obj.b;

}

else if(obj.a>obj.b)

{

cout<<"A is Maximum value:"<<obj.a;

}

else

{

cout<<"B is Maximum value:"<<obj.b;

}

}

int main()

{

A obj;

show(obj);

}

Templates

//1. Write a program of to swap the two values using template

#include<iostream>

using namespace std;

template <typename T>

T swap(T a,T b)

{

cout<<"put the value A:";

cin>>a;

cout<<"Put the value B:";

cin>>b;

a=a+b;

b=a-b;

a=a-b;

cout<<"After the swapping value of A:"<<a<<endl;

cout<<"After the swapping value of B:"<<b;

}

int main()

{

swap<int>(1,1);

swap<char>('a','b');

}

//2. Write a program of to sort the array using templates

#include <iostream>

using namespace std;

// A template function to implement bubble sort.

template <class T>

void bubbleSort(T a[], int n)

{

for (int i = 0; i < n - 1; i++)

for (int j = n - 1; i < j; j--)

if (a[j] < a[j - 1])

swap(a[j], a[j - 1]);

}

// Driver Code

int main()

{

int a[5] = { 10, 50, 30, 40, 20 };

int n = sizeof(a) / sizeof(a[0]);

// calls template function

bubbleSort<int>(a, n);

cout << " Sorted array : ";

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

cout << a[i] << " ";

cout << endl;

return 0;

}