**Assignment 2: Functions in C++**

1. Write a C++ program to calculate Compound Interest using concept of pass by reference and return by reference. Read principal amount, rate of interest and period from the user.

Sol

#include <iostream>

#include <cmath>

using namespace std;

// Function prototype

void calculateCompoundInterest(double &principal, double &rate, int &period, double &compoundInterest);

int main()

{

    double principal, rate, compoundInterest;

    int period;

    // Read principal amount, rate of interest, and period from the user

    cout << "Enter principal amount: ";

    cin >> principal;

    cout << "Enter rate of interest: ";

    cin >> rate;

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

    cin >> period;

    // Calculate compound interest

    calculateCompoundInterest(principal, rate, period, compoundInterest);

    // Display the compound interest

    cout << "Compound interest: " << compoundInterest << endl;

    return 0;

}

// Function to calculate compound interest

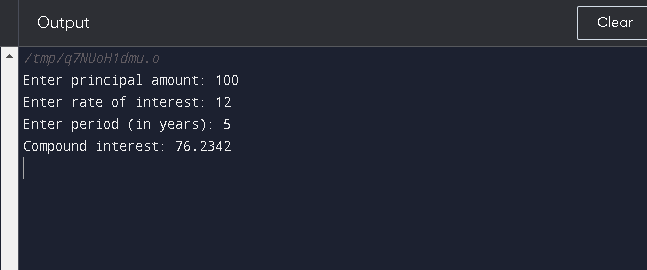
void calculateCompoundInterest(double &principal, double &rate, int &period, double &compoundInterest)

{

    compoundInterest = principal \* pow(1 + (rate / 100), period) - principal;

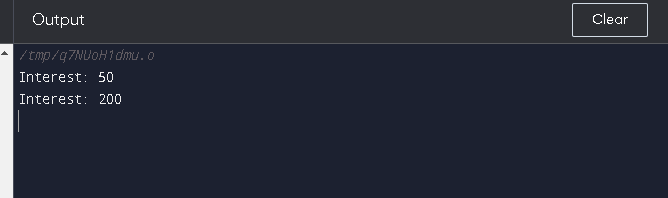
}

s

OUTPUT  


1. Write a function Interest(principal, int\_rate, year). Use function overloading and default arguments to set the values of int\_rate and year.
2. #include <iostream>
3. using namespace std;
4. double Interest(double principal, double int\_rate, int year)
5. {
6. // Calculate the interest
7. double interest = principal \* int\_rate \* year;
8. // Return the interest
9. return interest;
10. }
11. // Overloaded function with default arguments
12. double Interest(double principal, double int\_rate = 0.05)
13. {
14. int year=1;
15. // Call the original function with the default arguments
16. return Interest(principal, int\_rate, year);
17. }
18. int main()
19. {
20. // Calculate the interest on 1000 at 5% for 1 year
21. double interest1 = Interest(1000);
22. cout << "Interest: " << interest1 << endl;
23. // Calculate the interest on 1000 at 10% for 2 years
24. double interest2 = Interest(1000, 0.1, 2);
25. cout << "Interest: " << interest2 << endl;
26. return 0;
27. }

OUTPUT



1. Create a class Cone(radius, height). Declare a static member count that will store the number of objects created. Write a static function that display() the number of objects created. Write an area() function as inline function to display area of the cone.

Soln

#include <iostream>

#include <cmath>

using namespace std;

class Cone

{

private:

    double radius;

    double height;

    static int count; // static member variable to store the number of objects created

public:

    // Constructor to initialize radius and height of the cone

    Cone(double r, double h)

    {

        radius = r;

        height = h;

        ++count; // increment the count each time an object is created

    }

    // static function to display the number of objects created

    static void display()

    {

        cout << "Number of objects created: " << count << endl;

    }

    // inline function to calculate and display the surface area of the cone

    inline double area()

    {

        //double surface\_area = M\_PI \* radius \* (radius + sqrt(pow(radius, 2) + pow(height, 2)));

        //cout << "Surface area of the cone: " << surface\_area << endl;

        //double surface\_area = M\_PI \* radius \* (radius + sqrt(pow(radius, 2) + pow(height, 2)));

       // return surface\_area;

    }

};

// initialize the static member variable

int Cone::count = 0;

int main()

{

    // create two Cone objects

    Cone cone1(2, 3);

    Cone cone2(4, 5);

    // display the number of objects created

    Cone::display(); // Output: Number of objects created: 2

    // calculate and display the surface area of the first cone

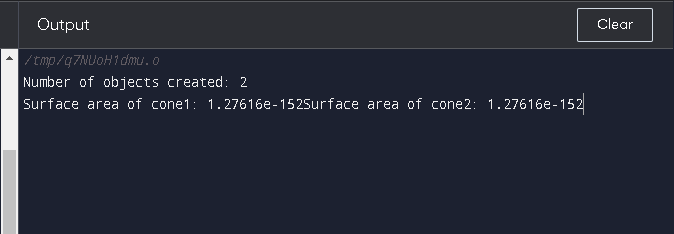
    cout<<"Surface area of cone1: "<<cone1.area(); // Output: Surface area of the cone: 37.699111843077515

    cout<<"Surface area of cone2: "<<cone2.area();

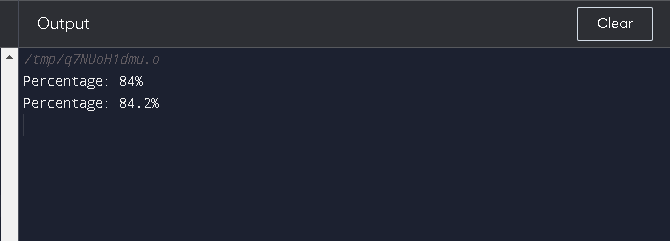
    return 0;

}

OUTPUT



1. Create two classes Academics(rollno, name, m1, m2,m3) and Curricular(c1,c2). Write a friend class result to calculate percentage of a student using academics and curricular marks.
2. #include <iostream>
3. #include <string>
4. using namespace std;
5. class Academics
6. {
7. private:
8. int rollno;
9. string name;
10. int m1, m2, m3;
11. public:
12. Academics(int rollno, string name, int m1, int m2, int m3)
13. : rollno(rollno), name(name), m1(m1), m2(m2), m3(m3)
14. { }
15. // Declare the Result class as a friend so that it can access the private members of the Academics class
16. friend class Result;
17. };
18. class Curricular
19. {
20. private:
21. int c1, c2;
22. public:
23. Curricular(int c1, int c2) : c1(c1), c2(c2)
24. { }
25. // Declare the Result class as a friend so that it can access the private members of the Curricular class
26. friend class Result;
27. };
28. class Result
29. {
30. public:
31. // Calculate the percentage for a student based on their marks in Academics and Curricular
32. static float percentage(const Academics &academics, const Curricular &curricular)
33. {
34. float total = academics.m1 + academics.m2 + academics.m3 + curricular.c1 + curricular.c2;
35. return (total / 500) \* 100;
36. }
37. };
38. int main()
39. {
40. Academics a(1, "John", 85, 90, 95);
41. Curricular c(80, 70);
42. float percentage = Result::percentage(a, c);
43. cout << "Percentage: " << percentage << "%" << endl;
45. Academics a1(2, "Wick", 85, 90, 95);
46. Curricular c1(81, 70);
47. float percentage1 = Result::percentage(a1, c1);
48. cout << "Percentage: " << percentage1 << "%" << endl;
49. return 0;
50. }



1. Create a class Account(Ano, Name, balance, int\_rate). Write a friend function Total\_bal() to calculate total interest earned on the balance.

#include <iostream>

#include <string>

using namespace std;

class Account

{

private:

    int A\_no;

    string Name;

    double balance;

    double int\_rate;

public:

    Account(int A, string N, double B, double IR) : A\_no(A), Name(N), balance(B), int\_rate(IR) {}

    friend double Total\_bal(const Account& a);

};

double Total\_bal(const Account& a)

{

    return a.balance \* a.int\_rate;

}

int main()

{

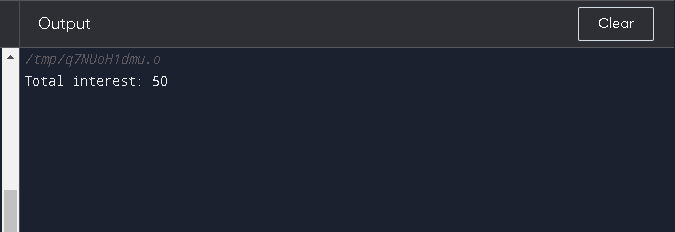
    Account a(12345, "John Smith", 1000.0, 0.05);

    cout << "Total interest: " << Total\_bal(a) << endl;

    return 0;

}

OUTPUT



1. Create a class Manager(id, name, department, designation). Use default, parameterized and copy constructors to initialize the members of the class.

#include <iostream>

using namespace std;

class Manager

{

private:

    int id;

    string name;

    string department;

    string designation;

public:

    Manager()

        : id(0), name(""), department(""), designation("") {}

    Manager(int ID, string Name, string Department, string Designation)

        : id(ID), name(Name), department(Department), designation(Designation) {}

    Manager(const Manager& m)

        : id(m.id), name(m.name), department(m.department), designation(m.designation) {}

    void print()

    {

        cout << "ID: " << id << endl;

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

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

        cout << "Designation: " << designation << endl;

        cout << endl;

    }

};

int main()

{

    Manager m1; // Uses default constructor

    Manager m2(12345, "John Smith", "IT", "Manager"); // Uses parameterized constructor

    Manager m3(m2); // Uses copy constructor

    m1.print();

    m2.print();

    m3.print();

    return 0;

}

