**14 programs**

**1-**Write a C++ program with a main function that prints "Hello, World!" to the console

#include<iostream>

using namespace std;

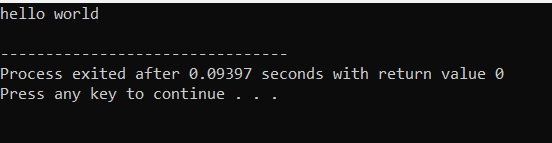
int main()

{

cout<<"hello world"<<endl;

return 0;

}



2-Create a C++ program with a function prototype for a function named **addNumbers** that takes two integers as parameters and returns their sum. Implement the function below the main function and use it to add two numbers.

#include<iostream>

using namespace std;

int add(int a,int b)

{

return a+b;

}

int main()

{

int num1,num2,sum;

cout<<"enter the number";

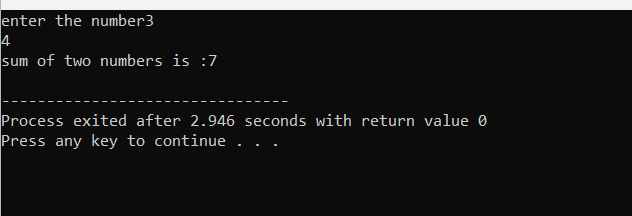
cin>>num1>>num2;

sum=add(num1,num2);

cout<<"sum of two numbers is :"<<sum<<endl;

return 0;

}



3-**Write a C++ program that demonstrates call by reference by swapping the values of two variables.**

#include<iostream>

using namespace std;

void swap(int &num1,int &num2)

{

int temp;

temp=num1;

num1=num2;

num2=temp;

}

int main()

{

int a,b;

cout<<"enter the numbers";

cin>>a>>b;

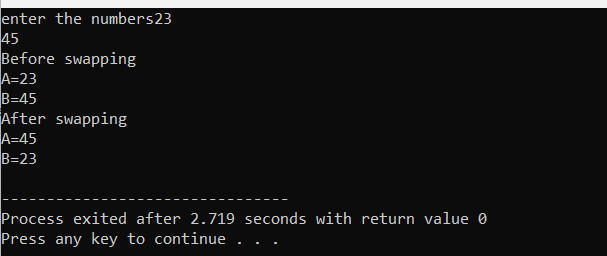
cout<<"Before swapping"<<"\nA="<<a<<"\nB="<<b<<endl;

swap(a,b);

cout<<"After swapping"<<"\nA="<<a<<"\nB="<<b<<endl;

return 0;

}



4-. **Develop a C++ program with a function that returns a reference to an integer variable and modifies its value. Use this function to update the value of a variable in the main function**

**#include <iostream>**

using namespace std;

int& modifyValue(int& x)

{

x += 10;

return x;

}

int main()

{

int num;

cout<<"enter the number";

cin>>num;

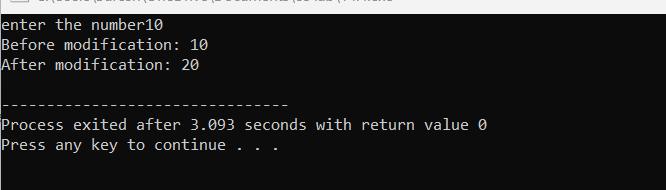
cout << "Before modification: " << num <<endl;

modifyValue(num);

cout << "After modification: " << num << endl;

return 0;

}



5**-. Implement an inline function named square that calculates the square of a number. Use this function to square a user-input integer.**

#include<iostream>

using namespace std;

class square

{

private:

int n,r;

float n1,r1;

public:

void input();

void calc();

void display();

};

inline void square::input()

{

cout<<"Enter an integer :: ";

cin>>n;

cout<<"\nEnter a float no. :: ";

cin>>n1;

}

inline void square::calc()

{

r=n\*n;

r1=n1\*n1;

}

inline void square::display()

{

cout<<"\nSquare of integer [ "<<n<<" ] = "<<r<<"\n";

cout<<"\nSquare of float [ "<<n1<<" ] = "<<r1<<"\n";

}

int main ()

{

square s;

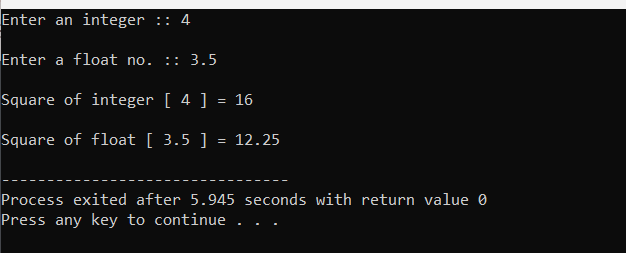
s.input();

s.calc();

s.display();

return 0;

}



6-**Write a C++ program that defines a function named printMessage with a default argument "Hello". Call this function without passing any argument**

#include <iostream>

#include <string>

using namespace std;

void printMessage(const string& message = "Hello") {

cout << message << endl;

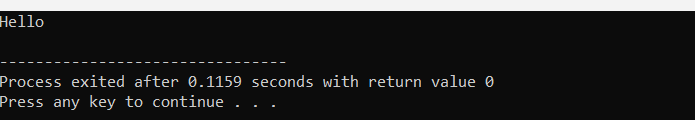
}

int main() {

printMessage();

return 0;

}



7-**Create a C++ program with two overloaded functions named area - one to calculate the area of a rectangle (length \* width) and another to calculate the area of a circle (π \* radius^2). Use function overloading to determine which function to call based on the number of arguments**.

#include <iostream>

#include <cmath>

double area(double length, double width) {

return length \* width;

}

double area(double radius) {

const double PI = 3.14159265358979323846;

return PI \* pow(radius, 2);

}

int main() {

double rectangleLength = 5.0;

double rectangleWidth = 3.0;

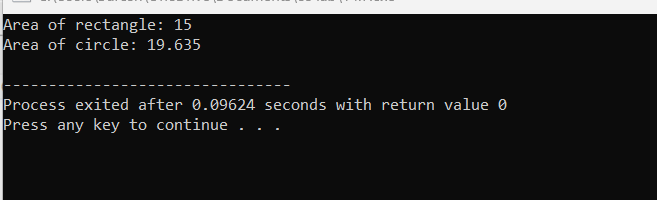
std::cout << "Area of rectangle: " << area(rectangleLength, rectangleWidth) << std::endl;

double circleRadius = 2.5;

std::cout << "Area of circle: " << area(circleRadius) << std::endl;

return 0;

}



8-**Define a C++ class named Rectangle with private member variables length and width. Implement public member functions to set and get the length and width of the rectangle.**

#include <iostream>

class Rectangle {

private:

double length;

double width;

public:

Rectangle() {

length = 0.0;

width = 0.0;

}

void setLength(double l) {

length = l;

}

void setWidth(double w) {

width = w;

}

double getLength() const {

return length;

}

double getWidth() const {

return width;

}

double calculateArea() const {

return length \* width;

}

};

int main() {

Rectangle rect;

rect.setLength(5.0);

rect.setWidth(3.0);

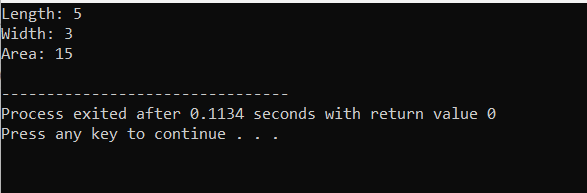
std::cout << "Length: " << rect.getLength() << std::endl;

std::cout << "Width: " << rect.getWidth() << std::endl;

std::cout << "Area: " << rect.calculateArea() << std::endl;

return 0;

}



9-Extend the Rectangle class to include an array of integers named coordinates to store the (x, y) coordinates of the rectangle. Implement member functions to set and get the coordinates

#include <iostream>

class Rectangle {

private:

double length;

double width;

int coordinates[2];

public:

Rectangle() {

length = 0.0;

width = 0.0;

coordinates[0] = 0;

coordinates[1] = 0;

}

void setLength(double l) {

length = l;

}

void setWidth(double w) {

width = w;

}

void setCoordinates(int x, int y) {

coordinates[0] = x;

coordinates[1] = y;

}

double getLength() const {

return length;

}

double getWidth() const {

return width;

}

int\* getCoordinates() {

return coordinates;

}

double calculateArea() const {

return length \* width;

}

};

int main() {

Rectangle rect;

rect.setLength(5.0);

rect.setWidth(3.0);

rect.setCoordinates(2, 3);

std::cout << "Length: " << rect.getLength() << std::endl;

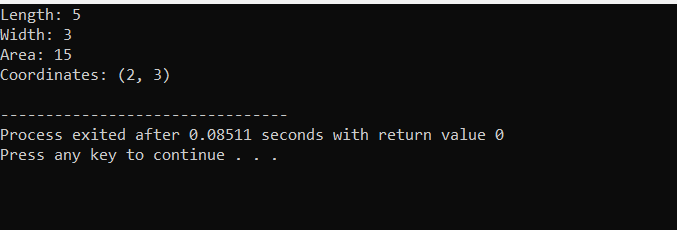
std::cout << "Width: " << rect.getWidth() << std::endl;

std::cout << "Area: " << rect.calculateArea() << std::endl;

std::cout << "Coordinates: (" << rect.getCoordinates()[0] << ", " << rect.getCoordinates()[1] << ")" << std::endl;

return 0;

}



10-Add a static member function named **countRectangles** to the **Rectangle** class that keeps track of the total number of rectangle objects created. Display the count in the main function.

#include <iostream>

class Rectangle {

private:

double length;

double width;

int coordinates[2];

static int totalRectangles;

public:

Rectangle() {

length = 0.0;

width = 0.0;

coordinates[0] = 0;

coordinates[1] = 0;

totalRectangles++;

}

~Rectangle() {

totalRectangles--;

}

void setLength(double l) {

length = l;

}

void setWidth(double w) {

width = w;

}

void setCoordinates(int x, int y) {

coordinates[0] = x;

coordinates[1] = y;

}

double getLength() const {

return length;

}

double getWidth() const {

return width;

}

int\* getCoordinates() {

return coordinates;

}

double calculateArea() const {

return length \* width;

}

static int countRectangles() {

return totalRectangles;

}

};

int Rectangle::totalRectangles = 0;

int main() {

Rectangle rect1;

Rectangle rect2;

Rectangle rect3;

rect1.setLength(5.0);

rect1.setWidth(3.0);

rect1.setCoordinates(2, 3);

std::cout << "Length: " << rect1.getLength() << std::endl;

std::cout << "Width: " << rect1.getWidth() << std::endl;

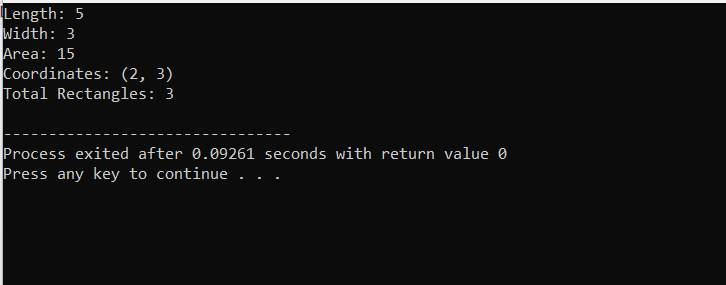
std::cout << "Area: " << rect1.calculateArea() << std::endl;

std::cout << "Coordinates: (" << rect1.getCoordinates()[0] << ", " << rect1.getCoordinates()[1] << ")" << std::endl;

std::cout << "Total Rectangles: " << Rectangle::countRectangles() << std::endl;

return 0;

}



11-**Write a C++ program that calculates the volume of a cube, rectangle, or cylinder based on user choice. Use function prototypes and function overloading to define separate functions for each shape's volume calculation.**

#include <iostream>

#include <cmath>

double calculateVolume(double side);

double calculateVolume(double length, double width, double height);

double calculateVolume(double radius, double height);

int main() {

char choice;

std::cout << "Choose a shape to calculate volume (c for cube, r for rectangle, or y for cylinder): ";

std::cin >> choice;

switch (choice) {

case 'c': {

double side;

std::cout << "Enter the side length of the cube: ";

std::cin >> side;

std::cout << "Volume of the cube: " << calculateVolume(side) << std::endl;

break;

}

case 'r': {

double length, width, height;

std::cout << "Enter the length, width, and height of the rectangle: ";

std::cin >> length >> width >> height;

std::cout << "Volume of the rectangle: " << calculateVolume(length, width, height) << std::endl;

break;

}

case 'y': {

double radius, height;

std::cout << "Enter the radius and height of the cylinder: ";

std::cin >> radius >> height;

std::cout << "Volume of the cylinder: " << calculateVolume(radius, height) << std::endl;

break;

}

default:

std::cout << "Invalid choice!" << std::endl;

}

return 0;

}

double calculateVolume(double side) {

return std::pow(side, 3);

}

double calculateVolume(double length, double width, double height) {

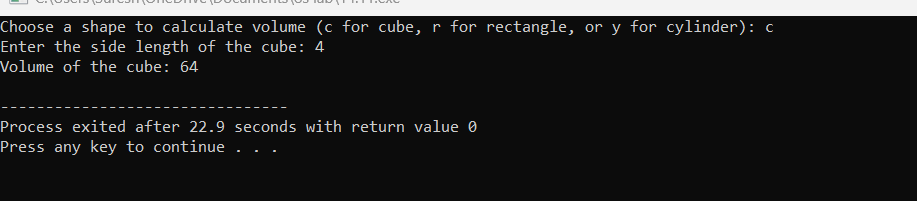
return length \* width \* height;

}

double calculateVolume(double radius, double height) {

return M\_PI \* std::pow(radius, 2) \* height;

}



12-**Define a class named Student with private member variables name, id, and an array grades to store the student's grades in three subjects. Implement member functions to set and get the student details and grades.**

#include <iostream>

#include <string>

using namespace std;

class Student {

private:

string name;

int id;

int grades[3];

public:

void setName(const string& n) {

name = n;

}

void setId(int studentId) {

id = studentId;

}

void setGrades(int grade1, int grade2, int grade3) {

grades[0] = grade1;

grades[1] = grade2;

grades[2] = grade3;

}

string getName() const {

return name;

}

int getId() const {

return id;

}

int\* getGrades() {

return grades;

}

};

int main() {

Student student;

student.setName("John Doe");

student.setId(12345);

student.setGrades(85, 90, 75);

cout << "Name: " << student.getName() << std::endl;

cout << "ID: " << student.getId() << std::endl;

int\* grades = student.getGrades();

std::cout << "Grades: ";

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

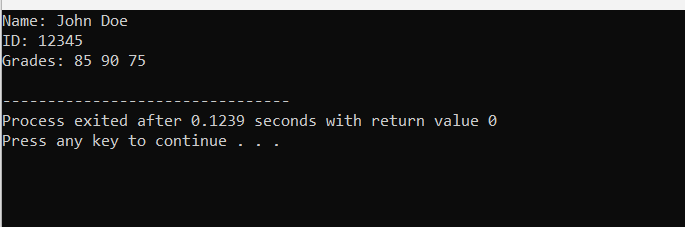
std::cout << grades[i] << " ";

}

std::cout << std::endl;

return 0;

}



13-**Create a C++ program that defines an inline function named calculateArea to calculate the area of a rectangle. Provide default arguments for length and width parameters. Use this function to calculate the area of a rectangle with user-input dimensions**

#include <iostream>

using namespace std;

inline double calculateArea(double length = 1.0, double width = 1.0) {

return length \* width;

}

int main() {

double length, width;

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

std::cin >> length;

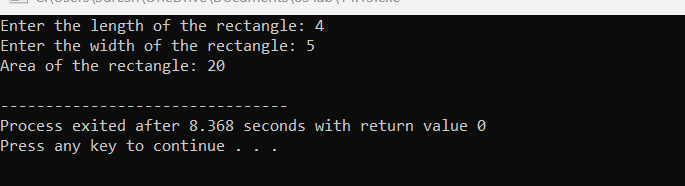
std::cout << "Enter the width of the rectangle: ";

std::cin >> width;

std::cout << "Area of the rectangle: " << calculateArea(length, width) << std::endl;

return 0;

}



14-. **Define a class named Employee with private member variables name and salary. Implement a static member function to calculate the average salary of an array of Employee objects.**

#include <iostream>

#include <string>

using namespace std;

class Employee {

private:

string name;

double salary;

public:

Employee(const

string& empName, double empSalary) : name(empName), salary(empSalary) {}

static double calculateAverageSalary(Employee\* employees, int numEmployees) {

if (numEmployees == 0)

return 0.0;

double totalSalary = 0.0;

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

totalSalary += employees[i].salary;

}

return totalSalary / numEmployees;

}

};

int main() {

Employee employees[3] = {

Employee("John", 50000),

Employee("Alice", 60000),

Employee("Bob", 70000)

};

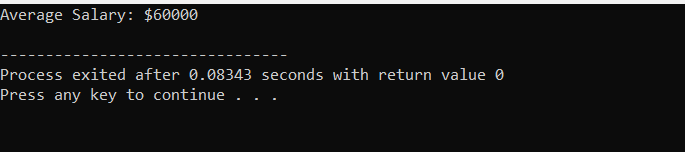
int numEmployees = sizeof(employees) / sizeof(employees[0]);

double averageSalary = Employee::calculateAverageSalary(employees, numEmployees);

std::cout << "Average Salary: $" << averageSalary << std::endl;

return 0;

}



15**. Define two classes, Complex and Matrix. Make Matrix a friend of Complex. Implement a function in Matrix class that multiplies a Complex number with a constant and returns the result by reference.**

#include<bits/stdc++.h>

using namespace std;

class Complex {

public:

int real, imaginary;

Complex(int tempReal = 0, int tempImaginary = 0)

{

real = tempReal;

imaginary = tempImaginary;

}

Complex addComp(Complex C1, Complex C2)

{

Complex temp;

temp.real = C1.real + C2.real;

temp.imaginary = C1.imaginary + C2.imaginary;

return temp;

}

};

int main()

{

Complex C1(3, 2);

cout<<"Complex number 1 : "<< C1.real

<< " + i"<< C1.imaginary<<endl;

Complex C2(9, 5);

cout<<"Complex number 2 : "<< C2.real

<< " + i"<< C2.imaginary<<endl;

Complex C3;

C3 = C3.addComp(C1, C2);

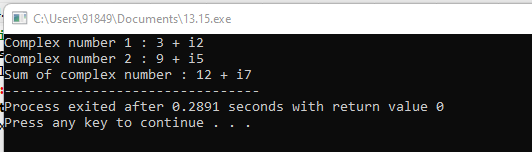
cout<<"Sum of complex number : "

<< C3.real << " + i"

<< C3.imaginary;

}

OUTPUT:



16. **Write a C++ program that contains overloaded functions named swap. Define one version that swaps two integers and another version that swaps two arrays of integers. Implement these functions using call by reference.**

#include <bits/stdc++.h>

using namespace std;

int main()

{

int a = 5, b = 10;

cout << "Before swapping a = " << a << " , b = " << b

<< endl;

swap(a, b);

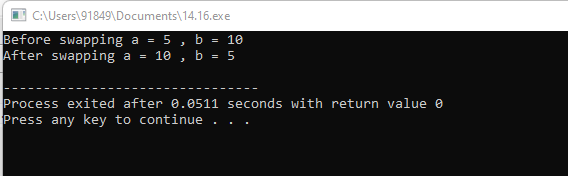
cout << "After swapping a = " << a << " , b = " << b

<< endl;

return 0;

}

OUTPUT:



17. **Define a class named Circle with private member variables radius and area. Implement member functions to set the radius, calculate the area, and display the circle's details.**

#include <iostream>

#include <cmath>

const double PI = 3.14159;

class Circle {

private:

double radius;

public:

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

double calculateArea() {

return PI \* pow(radius, 2);

}

double calculateCircumference() {

return 2 \* PI \* radius;

}

};

int main() {

double radius;

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

std::cin >> radius;

Circle circle(radius);

double area = circle.calculateArea();

std::cout << "Area: " << area << std::endl;

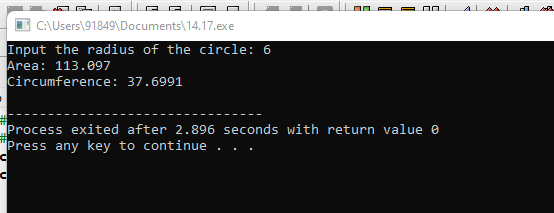
double circumference = circle.calculateCircumference();

std::cout << "Circumference: " << circumference << std::endl;

return 0;

}

OUTPUT:



18. **Create a C++ program that defines a class named Car with private member variables model, year, and price. Implement an array of Car objects and provide default values for year and price.**

#include <iostream>

#include <string>

class Car {

private:

std::string company;

std::string model;

int year;

public:

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

std::string getCompany() {

return company;

}

std::string getModel() {

return model;

}

int getYear() {

return year;

}

void setCompany(const std::string & comp) {

company = comp;

}

void setModel(const std::string & mdl) {

model = mdl;

}

void setYear(int yr) {

year = yr;

}

};

int main() {

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

std::cout << "Company: " << car.getCompany() << std::endl;

std::cout << "Model: " << car.getModel() << std::endl;

std::cout << "Year: " << car.getYear() << std::endl;

car.setCompany("BMW");

car.setModel("M4");

car.setYear(2022);

std::cout << "\nUpdated Company: " << car.getCompany() << std::endl;

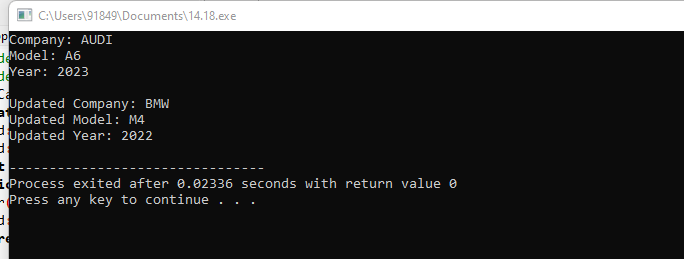
std::cout << "Updated Model: " << car.getModel() << std::endl;

std::cout << "Updated Year: " << car.getYear() << std::endl;

return 0;

}

OUTPUT:



19. Define a C++ function named **largestElement** that returns a reference to the largest element in an array of integers. Use this function to find the largest element in a user-input array.

#include <bits/stdc++.h>

using namespace std;

int largest(int arr[], int n)

{

int i;

int max = arr[0];

for (i = 1; i < n; i++)

if (arr[i] > max)

max = arr[i];

return max;

}

int main()

{

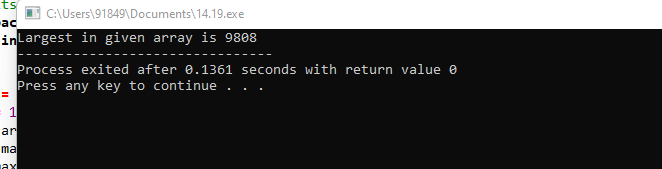
int arr[] = { 10, 324, 45, 90, 9808 };

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

cout << "Largest in given array is " << largest(arr, n);

return 0;

}

OUTPUT:  


20. Write a C++ program with function prototypes for calculating the area and perimeter of a rectangle. Implement these functions with default arguments for length and width. Prompt the user to enter the length and width to calculate the area and perimeter.

#include <iostream>

using namespace std;

int main()

{

int width, lngth, area, peri;

cout << "\n\n Find the Area and Perimeter of a Rectangle :\n";

cout << "-------------------------------------------------\n";

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

cin >> lngth;

cout << " Input the width of the rectangle : ";

cin >> width;

area = (lngth \* width);

peri = 2 \* (lngth + width);

cout << " The area of the rectangle is : "<< area << endl;

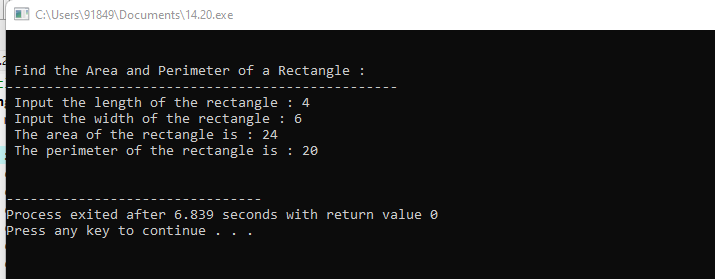
cout << " The perimeter of the rectangle is : "<< peri << endl;

cout << endl;

return 0;

}

OUTPUT:



21. Write a C++ program with function prototypes for calculating the area and perimeter of a rectangle, circle, and triangle. Implement overloaded functions to handle each shape's calculations.

#include<iostream>

using namespace std;

int area(int);

int area(int,int);

float area(float);

float area(float,float);

int main()

{

int s,l,b;

float r,bs,ht;

cout<<"Enter side of a square:";

cin>>s;

cout<<"Enter length and breadth of rectangle:";

cin>>l>>b;

cout<<"Enter radius of circle:";

cin>>r;

cout<<"Enter base and height of triangle:";

cin>>bs>>ht;

cout<<"Area of square is"<<area(s);

cout<<"\nArea of rectangle is "<<area(l,b);

cout<<"\nArea of circle is "<<area(r);

cout<<"\nArea of triangle is "<<area(bs,ht);

}

int area(int s)

{

return(s\*s);

}

int area(int l,int b)

{

return(l\*b);

}

float area(float r)

{

return(3.14\*r\*r);

}

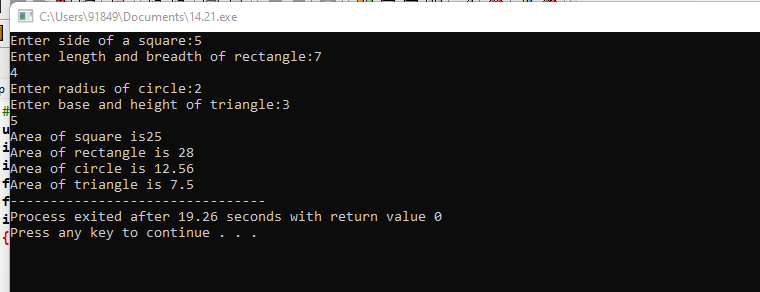
float area(float bs,float ht)

{

return((bs\*ht)/2);

}

OUTPUT:



22. **Define a class named Complex representing complex numbers with real and imaginary parts. Implement friend functions for addition, subtraction, multiplication, and division of complex numbers.**

#include <iostream>

class Complex {

private:

double real;

double imaginary;

public:

Complex(double real = 0, double imaginary = 0) : real(real), imaginary(imaginary) {}

friend Complex add(const Complex& c1, const Complex& c2);

friend Complex subtract(const Complex& c1, const Complex& c2);

friend Complex multiply(const Complex& c1, const Complex& c2);

friend Complex divide(const Complex& c1, const Complex& c2);

void display() const {

std::cout << real << " + " << imaginary << "i" << std::endl;

}

};

Complex add(const Complex& c1, const Complex& c2) {

return Complex(c1.real + c2.real, c1.imaginary + c2.imaginary);

}

Complex subtract(const Complex& c1, const Complex& c2) {

return Complex(c1.real - c2.real, c1.imaginary - c2.imaginary);

}

Complex multiply(const Complex& c1, const Complex& c2) {

double realPart = c1.real \* c2.real - c1.imaginary \* c2.imaginary;

double imaginaryPart = c1.real \* c2.imaginary + c1.imaginary \* c2.real;

return Complex(realPart, imaginaryPart);

}

Complex divide(const Complex& c1, const Complex& c2) {

double divisor = c2.real \* c2.real + c2.imaginary \* c2.imaginary;

double realPart = (c1.real \* c2.real + c1.imaginary \* c2.imaginary) / divisor;

double imaginaryPart = (c1.imaginary \* c2.real - c1.real \* c2.imaginary) / divisor;

return Complex(realPart, imaginaryPart);

}

int main() {

Complex c1(2, 3);

Complex c2(4, -2);

std::cout << "c1 = ";

c1.display();

std::cout << "c2 = ";

c2.display();

std::cout << "c1 + c2 = ";

add(c1, c2).display();

std::cout << "c1 - c2 = ";

subtract(c1, c2).display();

std::cout << "c1 \* c2 = ";

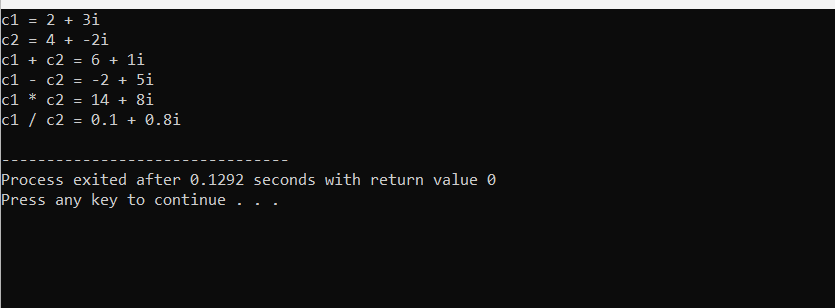
multiply(c1, c2).display();

std::cout << "c1 / c2 = ";

divide(c1, c2).display();

return 0;

}



23-