MODEL EXAMINATION

Name: T. Jithendra

Register Number: 192111521

Course Code: DSA0163

Course Name: Object Oriented

Programming with c++

SET – 05

1.write a program to read in an integer and determine if it is divisible by 3 or not using ternary conditional operator

#include<iostream>

using namespace std;

int main(){

int n;

cout << "enter a number: \n";

cin >> n;

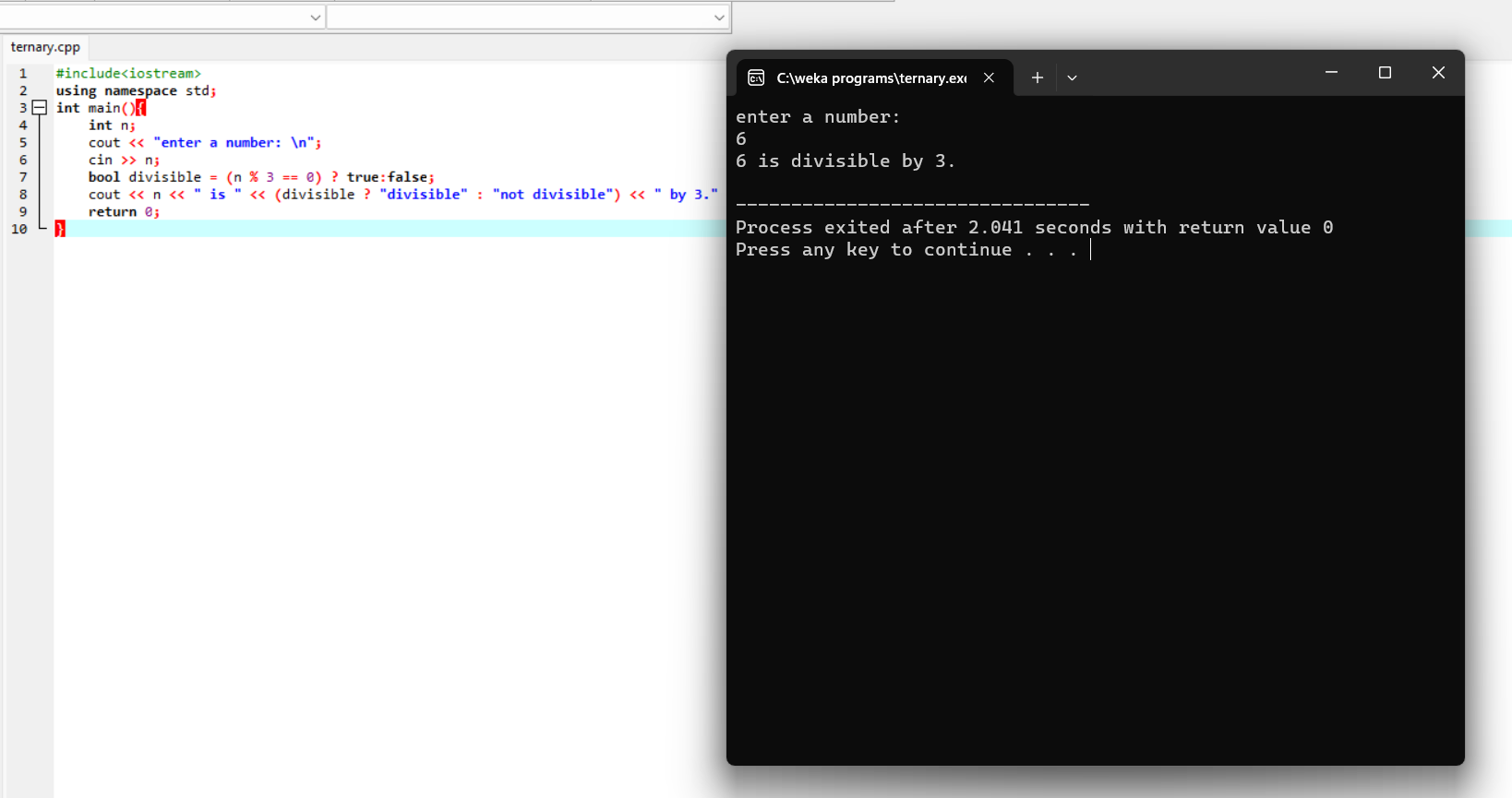
bool divisible = (n % 3 == 0) ? true:false;

cout << n << " is " << (divisible ? "divisible" : "not divisible") << " by 3." << endl;

return 0;

}

Output:



2.write a program to read in an integer and determine if it is a happy number or not.

#include<iostream>

using namespace std;

int main(){

int n;

cout << "enter a number: ";

cin >> n;

while (n!=1 and n!=4){

int sum = 0;

while (n > 0){

int rem = n % 10;

sum = sum + rem \* rem;

n = n / 10;

}

n = sum;

}

if (n==1){

cout << n << " is a happy number.";

}

else{

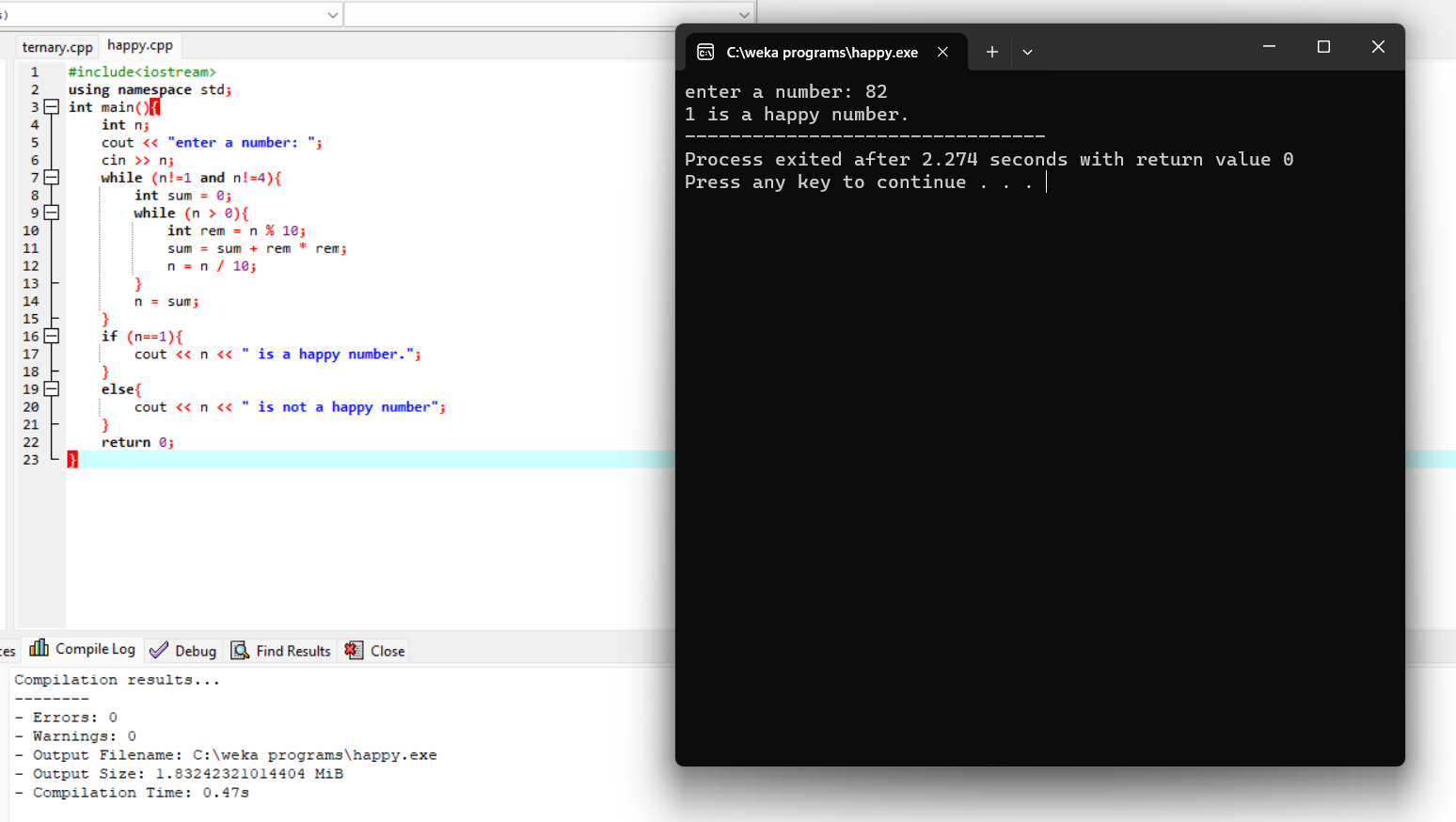
cout << n << " is not a happy number";

}

return 0;

}

Output:



3.write a program to print the following pattern using nested for loops:

1

232

34543

4567654

567898765

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter the number of rows: ";

cin >> n;

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

int num = i;

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

cout << " ";

}

for (int j = 1; j <= i; j++) {

cout << num;

num++;

}

num -= 2;

for (int j = 1; j <= i - 1; j++) {

cout << num;

num--;

}

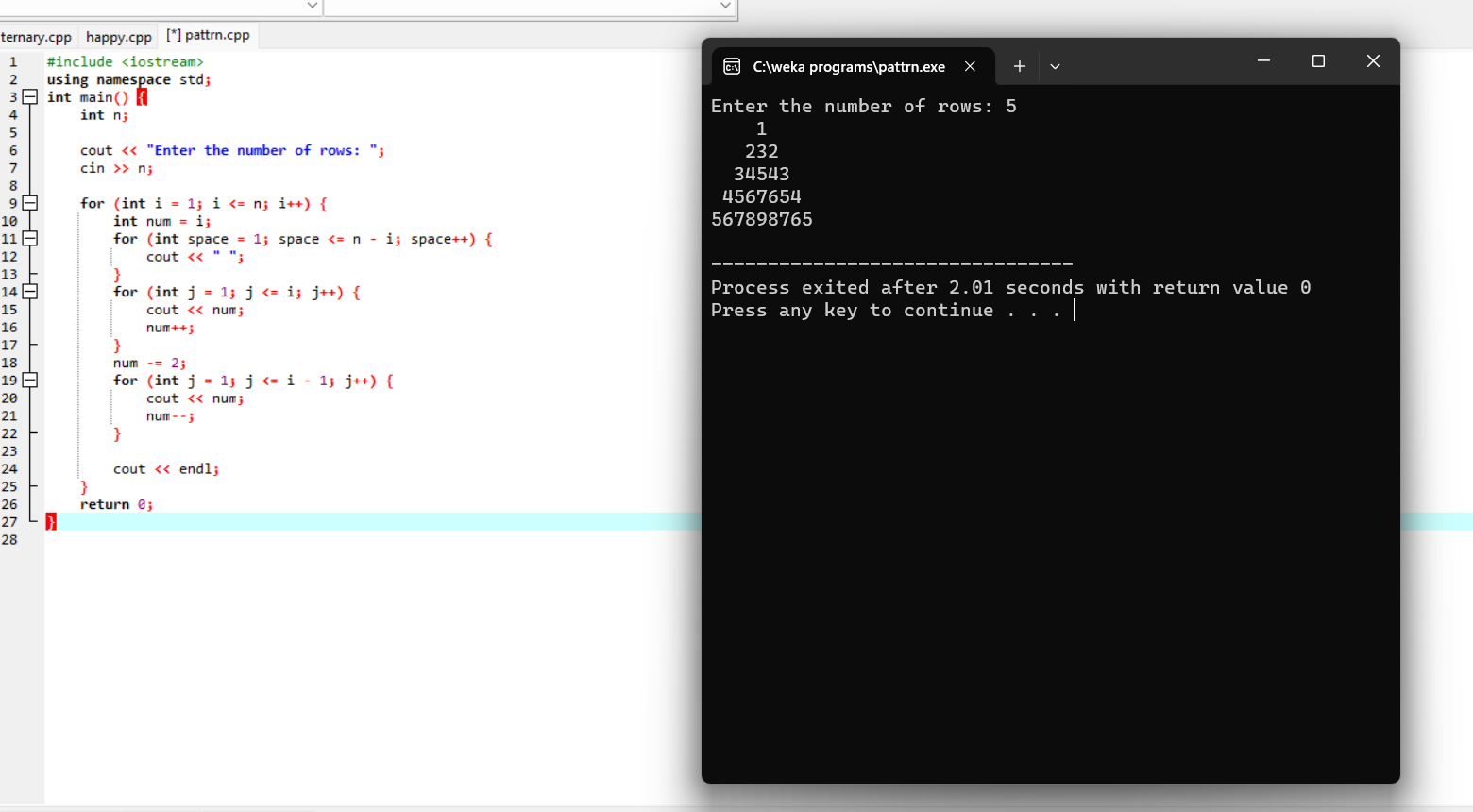
cout << endl;

}

return 0;

}

Output:



4.write a c++ program to create a dynamic two-dimensional array using pointers and display its values.

#include <iostream>

using namespace std;

int main() {

int numRows, numCols;

cout << "Enter the number of rows: ";

cin >> numRows;

cout << "Enter the number of columns: ";

cin >> numCols;

int\*\* dynamicArray = new int\*[numRows];

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

dynamicArray[i] = new int[numCols];

}

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

for (int j = 0; j < numCols; j++) {

cout << "Enter the value for dynamicArray[" << i << "][" << j << "]: ";

cin >> dynamicArray[i][j];

}

cout << endl;

}

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

for (int j = 0; j < numCols; j++) {

cout << dynamicArray[i][j] << " ";

}

cout << std::endl;

}

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

delete[] dynamicArray[i];

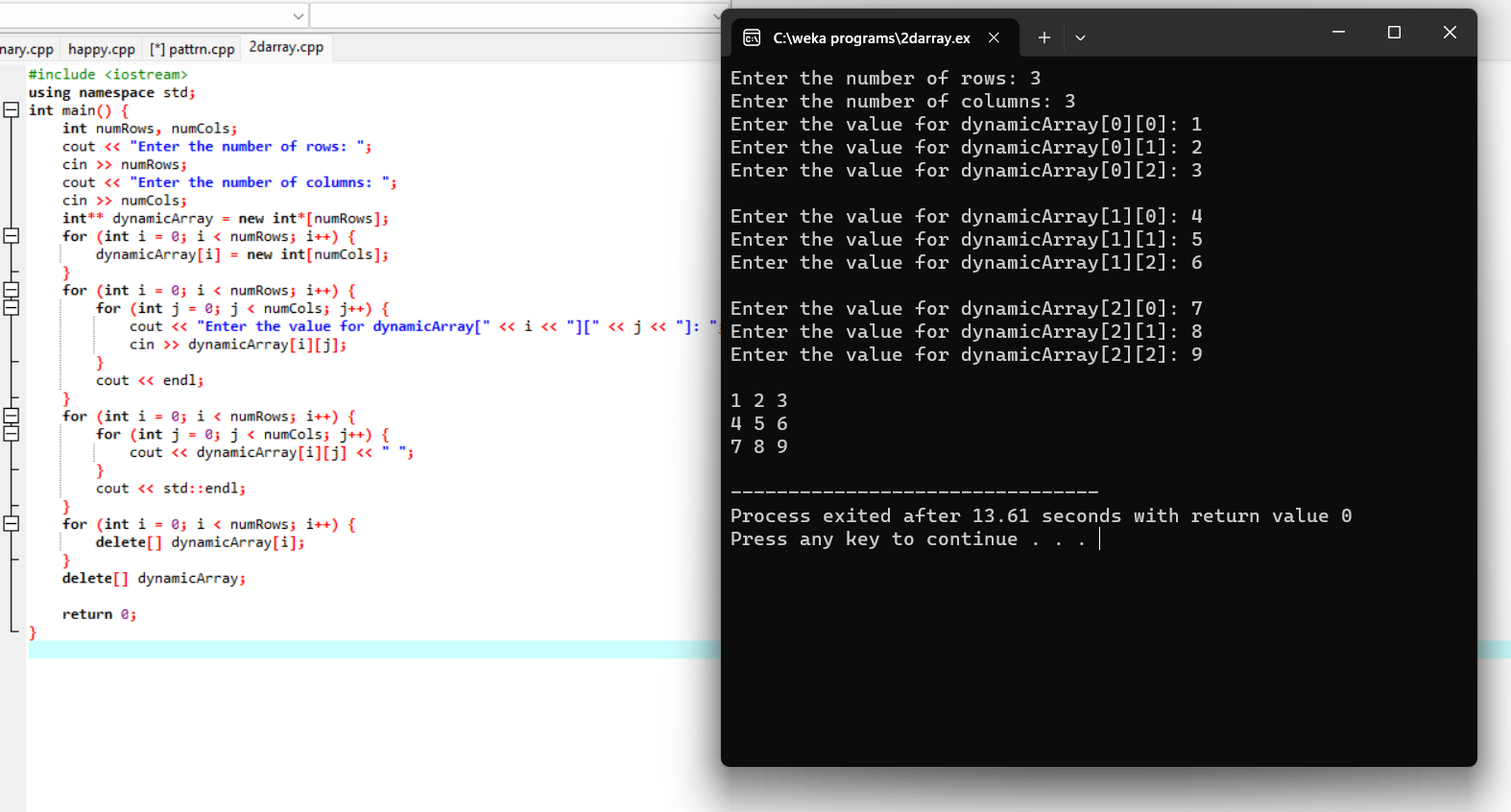
}

delete[] dynamicArray;

return 0;

}

Output:



5. #include <iostream>

#include <cmath>

using namespace std;

class Shape {

public:

virtual double area() const = 0;

virtual double volume() const = 0;

};

class Sphere : public Shape {

private:

double radius;

public:

Sphere(double r) : radius(r) {}

double area() const override {

return 4 \* M\_PI \* radius \* radius;

}

double volume() const override {

return (4.0 / 3.0) \* M\_PI \* radius \* radius \* radius;

}

};

class Cylinder : public Shape {

private:

double radius;

double height;

public:

Cylinder(double r, double h) : radius(r), height(h) {}

double area() const override {

return 2 \* M\_PI \* radius \* (radius + height);

}

double volume() const override {

return M\_PI \* radius \* radius \* height;

}

};

int main() {

Sphere sphere(3.0);

Cylinder cylinder(2.0, 4.0);

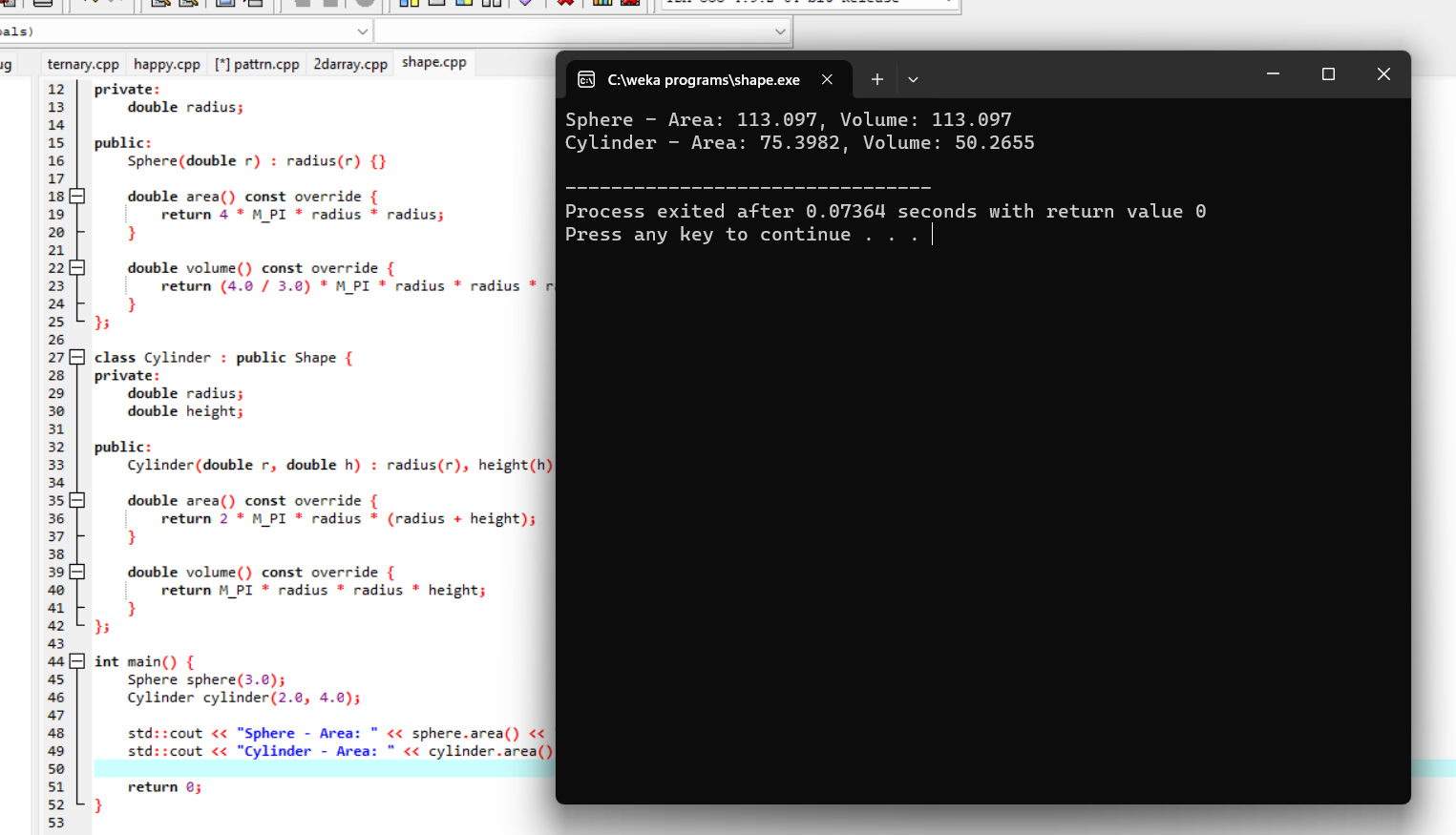
std::cout << "Sphere - Area: " << sphere.area() << ", Volume: " << sphere.volume() << std::endl;

std::cout << "Cylinder - Area: " << cylinder.area() << ", Volume: " << cylinder.volume() << std::endl;

return 0;

}

Output:



6. #include <iostream>

class Vehicle {

protected:

string make;

string model;

int year;

public:

Vehicle(const std::string& make, const std::string& model, int year)

: make(make), model(model), year(year) {}

virtual void Print() const {

cout << "Make: " << make << endl;

cout << "Model: " << model << endl;

cout << "Year: " << year << endl;

}

const string& GetMake() const { return make; }

void SetMake(const string& make) { this->make = make; }

const string& GetModel() const { return model; }

void SetModel(const string& model) { this->model = model; }

int GetYear() const { return year; }

void SetYear(int year) { this->year = year; }

};

class Car : public Vehicle {

private:

int seatingCapacity;

string fuelType;

public:

Car(const string& make, const string& model, int year,

int seatingCapacity, const string& fuelType)

: Vehicle(make, model, year), seatingCapacity(seatingCapacity),

fuelType(fuelType) {}

void Print() const override {

Vehicle Print();

cout << "Seating capacity: " << seatingCapacity << std::endl;

cout << "Fuel type: " << fuelType << std::endl;

}

int GetSeatingCapacity() const { return seatingCapacity; }

void SetSeatingCapacity(int seatingCapacity) {

this->seatingCapacity = seatingCapacity;

}

const std::string& GetFuelType() const { return fuelType; }

void SetFuelType(const std::string& fuelType) { this->fuelType = fuelType; }

};

class Truck : public Vehicle {

private:

int payloadCapacity;

int towingCapacity;

public:

Truck(const std::string& make, const std::string& model, int year,

int payloadCapacity, int towingCapacity)

: Vehicle(make, model, year), payloadCapacity(payloadCapacity),

towingCapacity(towingCapacity) {}

void Print() const override {

Vehicle::Print();

std::cout << "Payload capacity: " << payloadCapacity << std::endl;

std::cout << "Towing capacity: " << towingCapacity << std::endl;

}

int GetPayloadCapacity() const { return payloadCapacity; }

void SetPayloadCapacity(int payloadCapacity) {

this->payloadCapacity = payloadCapacity;

}

int GetTowingCapacity() const { return towingCapacity; }

void SetTowingCapacity(int towingCapacity) {

this->towingCapacity = towingCapacity;

}

};

int main() {

Car car("Toyota", "Camry", 2023, 5, "Gasoline");

Truck truck("Ford", "F-150", 2023, 1000, 5000);

std::cout << "Car:" << std::endl;

car.Print();

std::cout << std::endl;

std::cout << "Truck:" << std::endl;

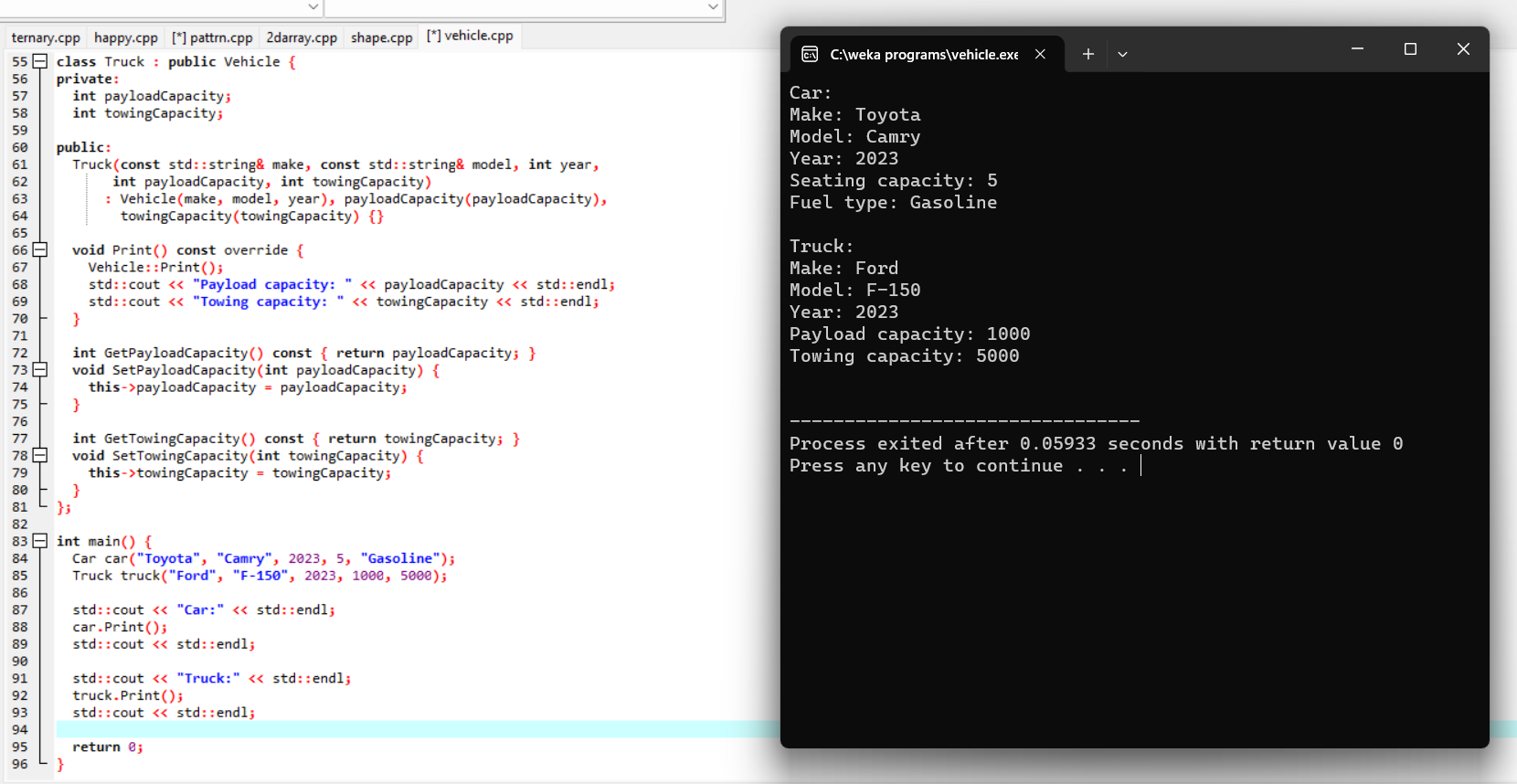
truck.Print();

std::cout << std::endl;

return 0;

}

Ouput:



7.write a c++ program to overload a function to calculate the area of a square, a rectangle, and a circle separately.

#include <iostream>

#include <cmath>

using namespace std;

const double PI = 3.14159265358979323846;

double calculateArea(double side) {

return side \* side;

}

double calculateArea(double length, double width) {

return length \* width;

}

double calculateArea(double radius) {

return PI \* radius \* radius;

}

int main() {

double side, length, width, radius;

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

cin >> side;

cout << "Area of the square: " << calculateArea(side) << std::endl;

cout << "Enter the length and width of a rectangle: ";

cin >> length >> width;

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

cout << "Enter the radius of a circle: ";

cin >> radius;

cout << "Area of the circle: " << calculateArea(radius) << std::endl;

return 0;

}

8.write a c++ program to create a class for a car with a constructor and destructor.

#include <iostream>

using namespace std;

class Car {

public:

Car(const std::string& make, const std::string& model, int year) {

this->make = make;

this->model = model;

this->year = year;

cout << "Car object created." << endl;

}

~Car() {

cout << "Car object destroyed." << endl;

}

void displayInfo() {

cout << "Make: " << make << ", Model: " << model << ", Year: " << year << endl;

}

private:

string make;

string model;

int year;

};

int main() {

Car myCar("Toyota", "Camry", 2020);

myCar.displayInfo();

return 0;

}

Output:

