**Assignment::1**

1.) Illustrate the following in two ways of object

notations.

a) Employee

b) Teacher

c) Car

Identify the object of the following class and

create object.

a) Company

b) Class

c) Vehicle

**program**

#include <iostream>

#include <string>

class Company {

public:

std::string name;

int employees;

};

class Class {

public:

std::string className;

int students;

};

class Vehicle {

public:

std::string make;

std::string model;

int year;

};

int main() {

Company companyObj;

companyObj.name = "ABC Corp";

companyObj.employees = 100;

Class classObj;

classObj.className = "Mathematics";

classObj.students = 30;

Vehicle vehicleObj;

vehicleObj.make = "Toyota";

vehicleObj.model = "Corolla";

vehicleObj.year = 2021;

std::cout << "Company: " << companyObj.name << ", Employees: " << companyObj.employees << std::endl;

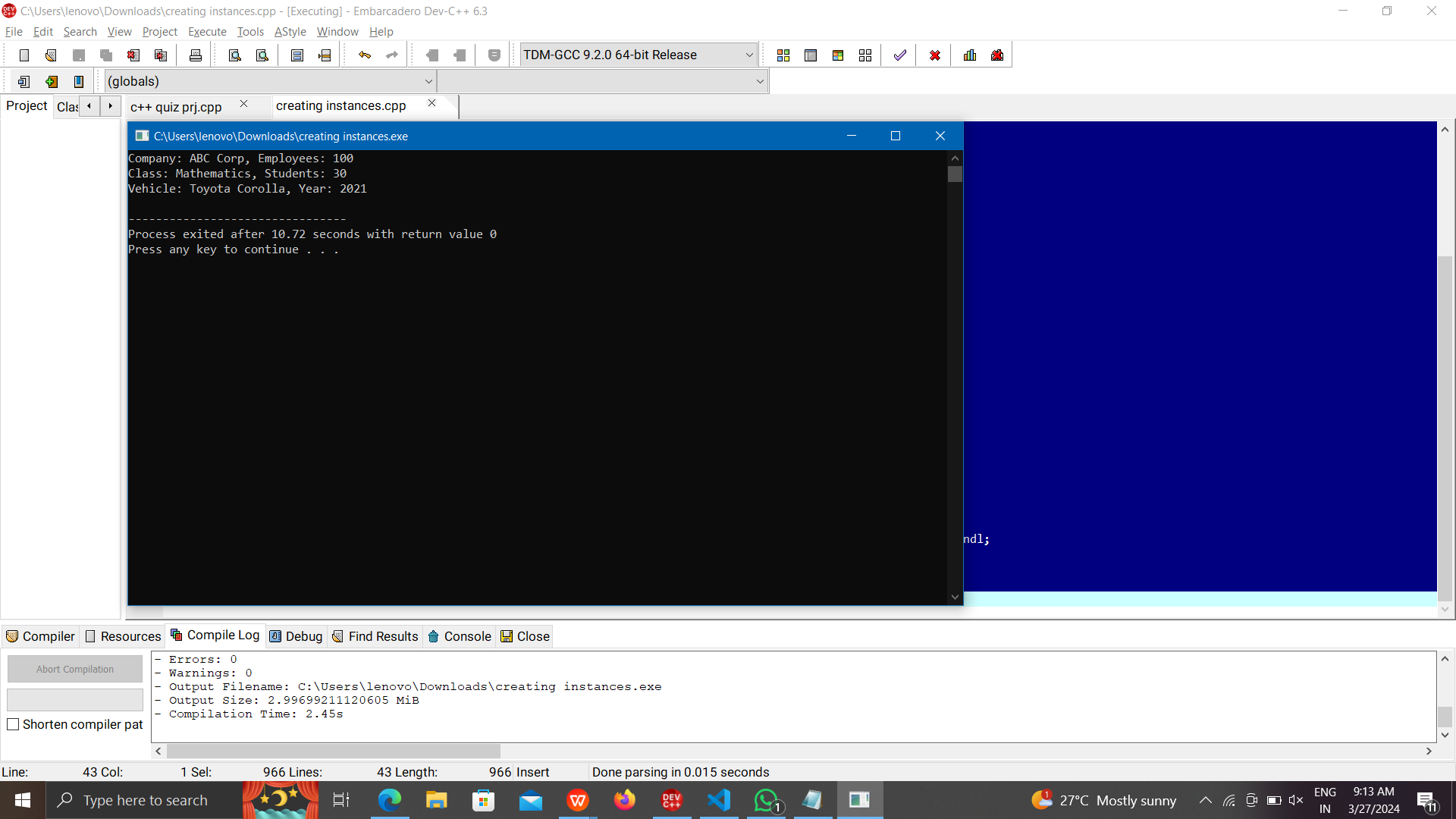
std::cout << "Class: " << classObj.className << ", Students: " << classObj.students << std::endl;

std::cout << "Vehicle: " << vehicleObj.make << " " << vehicleObj.model << ", Year: " << vehicleObj.year << std::endl;

return 0;

}

**Output:**



(2)Design a class hierarchy to represent a geometric shapes system. Include classes for circles, rectangles, and triangles. Implement methods for calculating the area and perimeter of each shape. Demonstrate the use of abstract classes and pure virtual functions. Discuss how this design supports future additions of new shapes.

**Program**:

#include <iostream>

class Shape {

public:

virtual float calculateArea() = 0;

virtual float calculatePerimeter() = 0;

};

class Circle : public Shape {

private:

float radius;

public:

Circle(float r) : radius(r) {}

float calculateArea() override {

return 3.14159 \* radius \* radius;

}

float calculatePerimeter() override {

return 2 \* 3.14159 \* radius;

}

};

class Rectangle : public Shape {

private:

float length;

float width;

public:

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

float calculateArea() override {

return length \* width;

}

float calculatePerimeter() override {

return 2 \* (length + width);

}

};

class Triangle : public Shape {

private:

float base;

float height;

float side1, side2, side3;

public:

Triangle(float b, float h, float s1, float s2, float s3) : base(b), height(h), side1(s1), side2(s2), side3(s3) {}

float calculateArea() override {

return 0.5 \* base \* height;

}

float calculatePerimeter() override {

return side1 + side2 + side3;

}

};

int main() {

Circle c(5);

Rectangle r(4, 6);

Triangle t(3, 4, 5, 4, 3);

std::cout << "Circle Area: " << c.calculateArea() << ", Perimeter: " << c.calculatePerimeter() << std::endl;

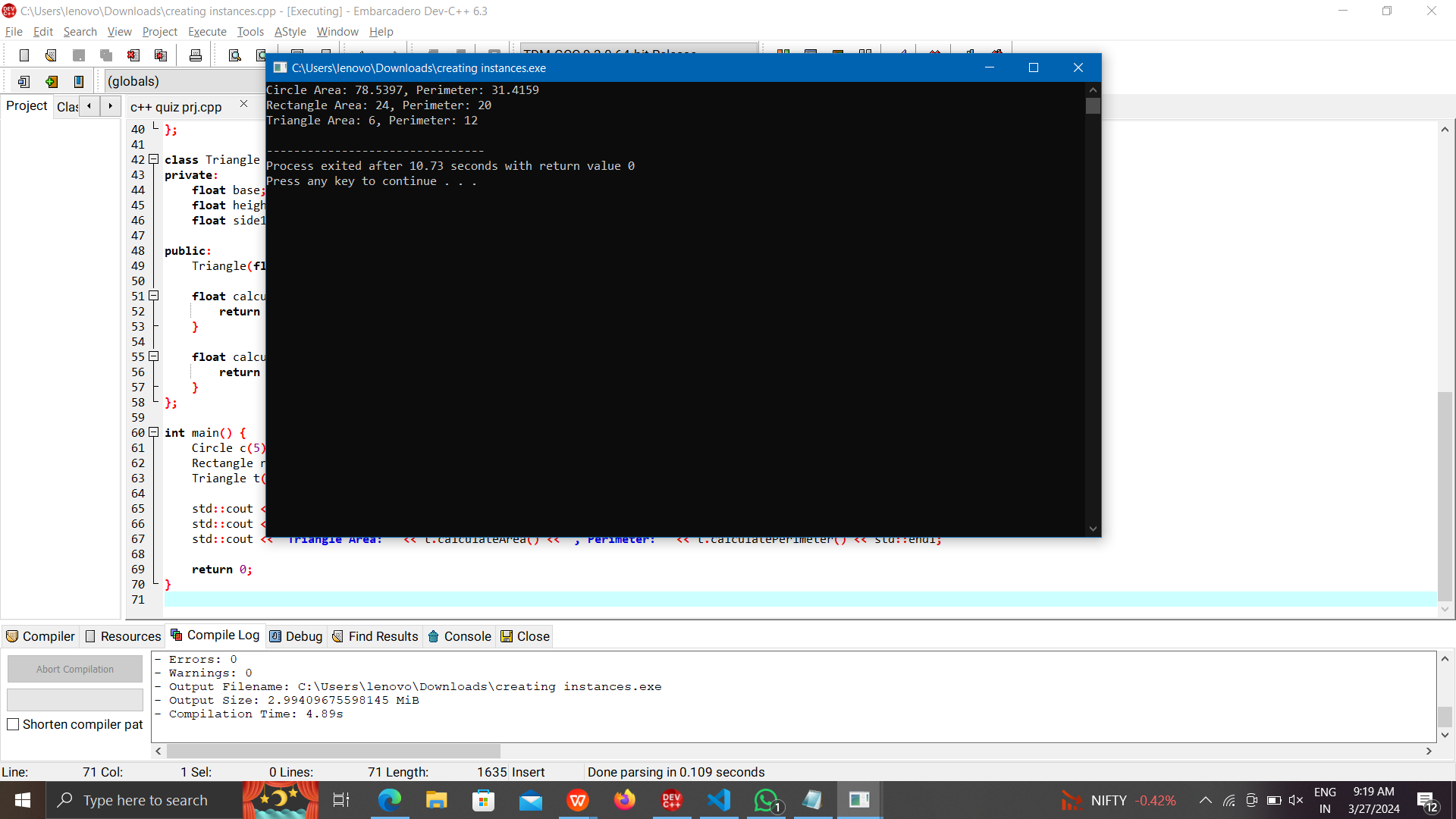
std::cout << "Rectangle Area: " << r.calculateArea() << ", Perimeter: " << r.calculatePerimeter() << std::endl;

std::cout << "Triangle Area: " << t.calculateArea() << ", Perimeter: " << t.calculatePerimeter() << std::endl;

return 0;

}

**Output:**



(3). Design a class hierarchy to represent a university system. Include classes for students, professors, and courses. Use appropriate OOP

principles like encapsulation, inheritance, and polymorphism. Provide a brief explanation of how your design promotes code reusability

and flexibility.

**Program**

#include <iostream>

#include <string>

#include <vector>

class Person {

protected:

std::string name;

int age;

public:

Person(const std::string& n, int a) : name(n), age(a) {}

std::string getName() const { return name; }

int getAge() const { return age; }

};

class Student : public Person {

private:

std::string studentID;

std::string major;

public:

Student(const std::string& n, int a, const std::string& id, const std::string& m) : Person(n, a), studentID(id), major(m) {}

void enrollCourse() {

std::cout << name << " has enrolled in a course." << std::endl;

}

void dropCourse() {

std::cout << name << " has dropped a course." << std::endl;

}

};

class Professor : public Person {

private:

std::string employeeID;

std::string department;

public:

Professor(const std::string& n, int a, const std::string& id, const std::string& dept) : Person(n, a), employeeID(id), department(dept) {}

void teachCourse() {

std::cout << name << " is teaching a course." << std::endl;

}

void gradeStudents() {

std::cout << name << " is grading students." << std::endl;

}

};

class Course {

private:

std::string courseCode;

std::string title;

Professor\* professor;

std::vector<Student\*> students;

public:

Course(const std::string& code, const std::string& t, Professor\* prof) : courseCode(code), title(t), professor(prof) {}

void addStudent(Student\* student) {

students.push\_back(student);

std::cout << student->getName() << " has been added to the course." << std::endl;

}

void removeStudent(Student\* student) {

}

};

int main() {

Professor prof("Dr. Smith", 45, "P123", "Computer Science");

Student student("Alice", 20, "S456", "Computer Science");

Course course("CS101", "Introduction to Computer Science", &prof);

course.addStudent(&student);

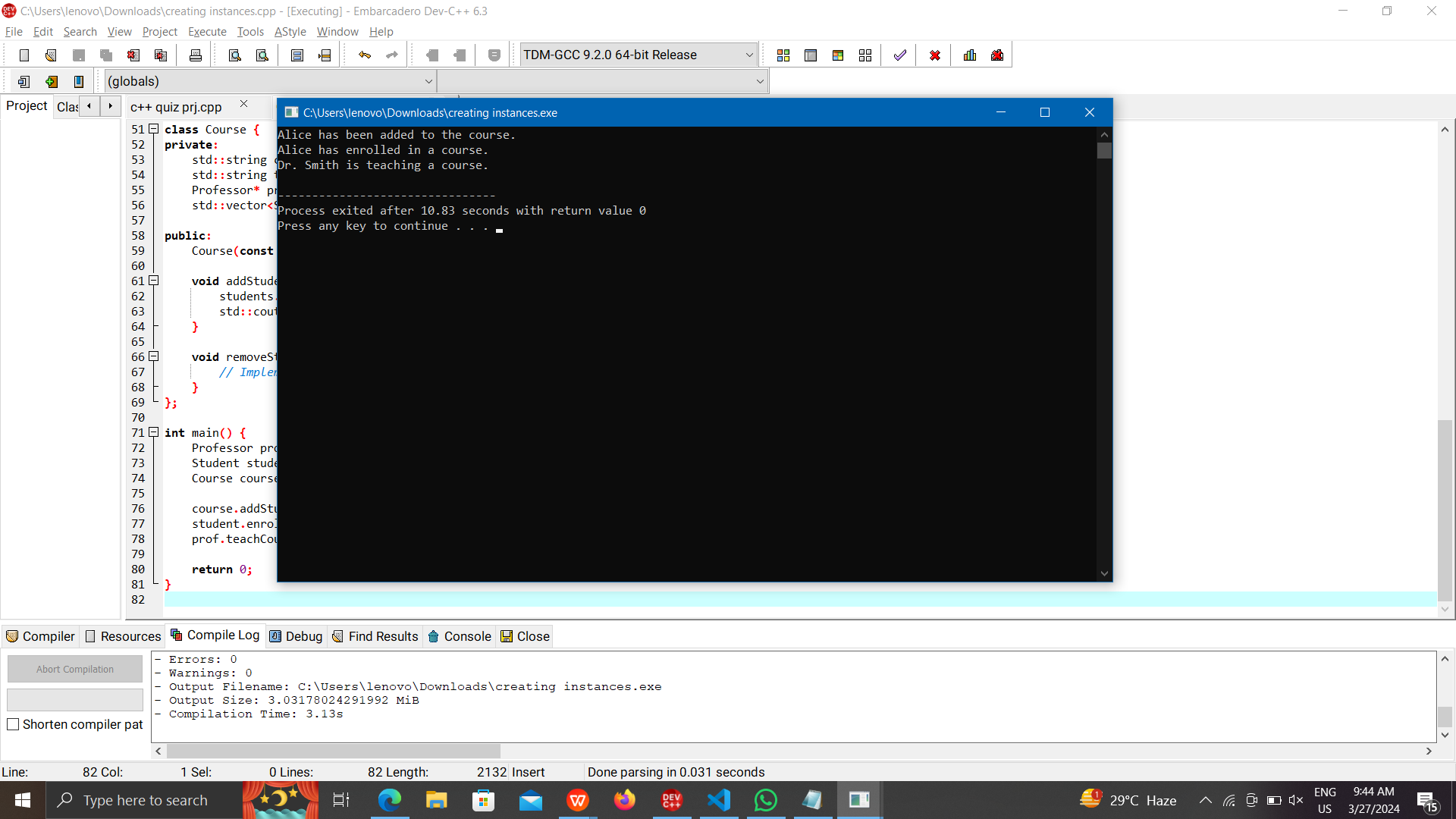
student.enrollCourse();

prof.teachCourse();

return 0;

}

**Output:**



(4).Design a class hierarchy to represent a geometric shapes system. Include classes for circles, rectangles, and triangles. Implement methods for calculating the area and perimeter of each shape. Demonstrate the use of abstract classes and pure virtual functions. Discuss how this design supports future additions of new shapes.

**Program:**

#include <iostream>

class Shape {

public:

virtual float calculateArea() = 0;

virtual float calculatePerimeter() = 0;

};

class Circle : public Shape {

private:

float radius;

public:

Circle(float r) : radius(r) {}

float calculateArea() override {

return 3.14159 \* radius \* radius;

}

float calculatePerimeter() override {

return 2 \* 3.14159 \* radius;

}

};

class Rectangle : public Shape {

private:

float length;

float width;

public:

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

float calculateArea() override {

return length \* width;

}

float calculatePerimeter() override {

return 2 \* (length + width);

}

};

class Triangle : public Shape {

private:

float base;

float height;

float side1, side2, side3;

public:

Triangle(float b, float h, float s1, float s2, float s3) : base(b), height(h), side1(s1), side2(s2), side3(s3) {}

float calculateArea() override {

return 0.5 \* base \* height;

}

float calculatePerimeter() override {

return side1 + side2 + side3;

}

};

int main() {

Circle c(5);

Rectangle r(4, 6);

Triangle t(3, 4, 5, 4, 3);

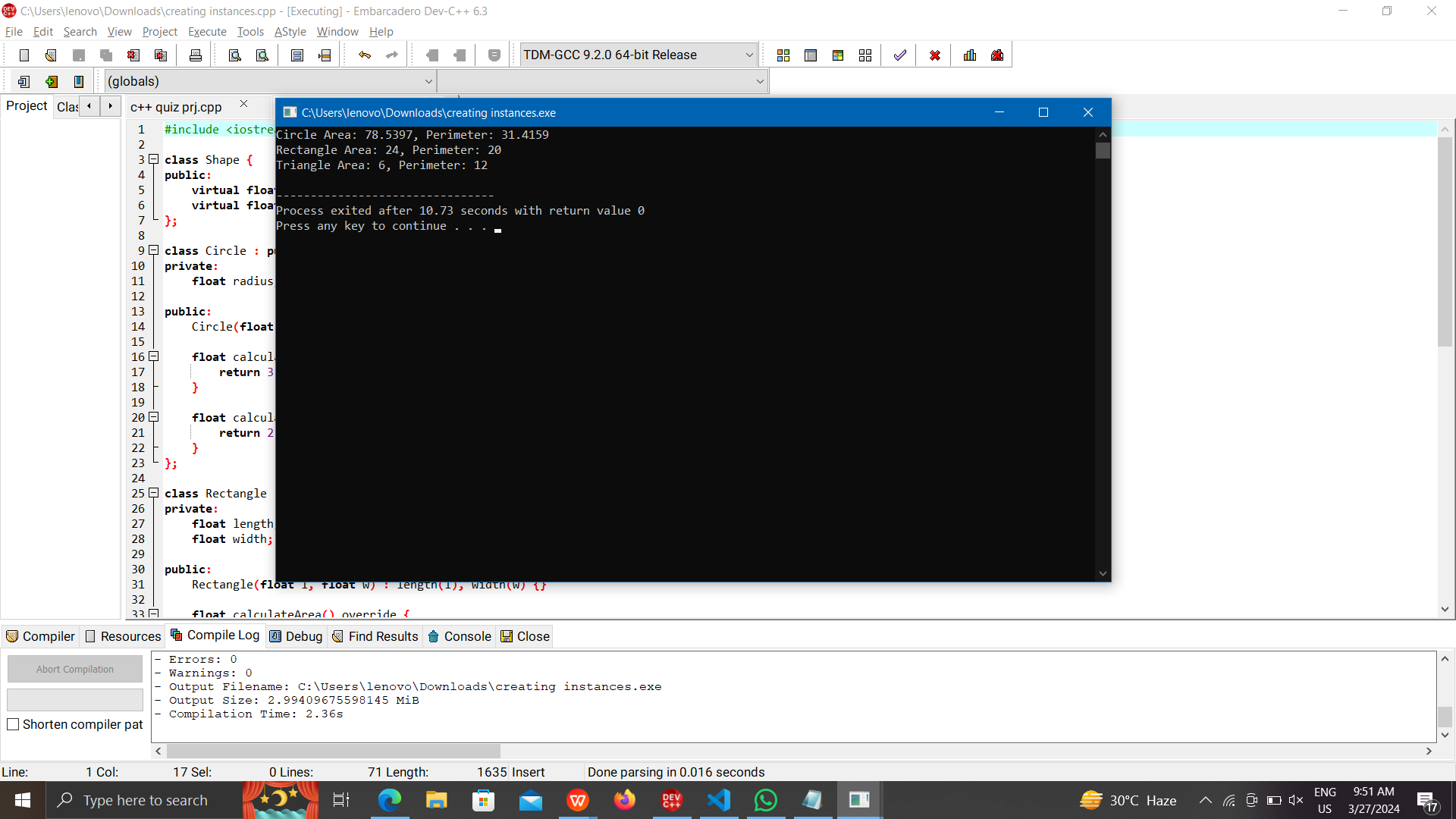
std::cout << "Circle Area: " << c.calculateArea() << ", Perimeter: " << c.calculatePerimeter() << std::endl;

std::cout << "Rectangle Area: " << r.calculateArea() << ", Perimeter: " << r.calculatePerimeter() << std::endl;

std::cout << "Triangle Area: " << t.calculateArea() << ", Perimeter: " << t.calculatePerimeter() << std::endl;

return 0;

}

**Output**

(5).In an organization they decide to give bonus

to all the employees on New Year. A 5%

bonus on salary is given to the grade A

workers and 10% bonus on salary to the grade

B workers. Write a program to enter the salary

and grade of the employee. If the salary of the

employee is less than $10,000 then the

employee gets an extra 2% bonus on salary

Calculate the bonus that has to be given to the

employee and print the salary that the

employee will get.

Sample Input & Output:

Enter the grade of the employee: B

Enter the employee salary: 50000

Salary=50000

Bonus=5000.0

Total to be paid: 55000.0

**Program**

#include <iostream>

int main() {

char grade;

double salary, bonus = 0.0;

std::cout << "Enter the grade of the employee: ";

std::cin >> grade;

std::cout << "Enter the employee salary: ";

std::cin >> salary;

if (grade == 'A') {

bonus = 0.05 \* salary;

} else if (grade == 'B') {

bonus = 0.10 \* salary;

}

if (salary < 10000) {

bonus += 0.02 \* salary;

}

double totalSalary = salary + bonus;

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

std::cout << "Bonus=" << bonus << std::endl;

std::cout << "Total to be paid: " << totalSalary << std::endl;

return 0;

}

**Output:**

