**Assignment 4**

(1). Write a C++ program that demonstrates the use of pointers to

change the local values of variables defined in one function from

within another function. Create two functions, **main()** and

**modifyValues()**. The **main()** function should declare local

variables, and the **modifyValues()** function should take pointers

as parameters to modify the values of these local variables.

**Program:**

#include <iostream>

void modifyValues(int\* a, int\* b) {

\*a = 20;

\*b = 30;

}

int main() {

int x = 10;

int y = 15;

std::cout << "Before calling modifyValues():" << std::endl;

std::cout << "x = " << x << ", y = " << y << std::endl;

modifyValues(&x, &y);

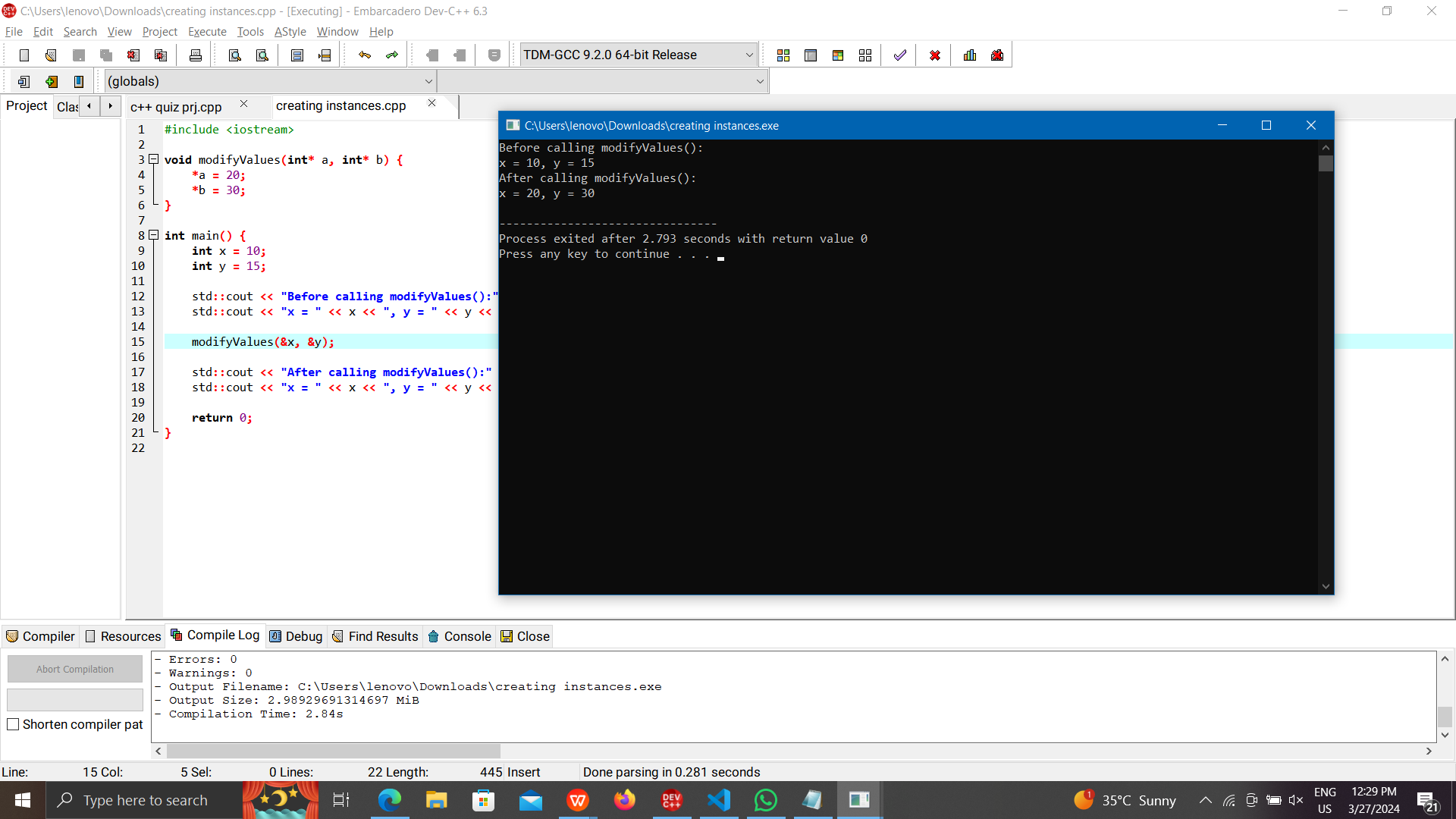
std::cout << "After calling modifyValues():" << std::endl;

std::cout << "x = " << x << ", y = " << y << std::endl;

return 0;

**}**

**Output:**



(2). Create a C++ program that utilizes multiple inheritance to

display employee information. Design a class hierarchy

involving three classes: **Person, Employee**, and **DisplayInfo**.

The Person class should contain basic information such as name

and address, the **Employee** class should include details specific

to employment, such as employee ID and salary, and the

**DisplayInfo** class should handle the functionality to display the

information.

**Program:**

#include <iostream>

#include <string>

class Person {

protected:

std::string name;

std::string address;

public:

Person(const std::string& n, const std::string& addr) : name(n), address(addr) {}

};

class Employee {

protected:

int employeeID;

double salary;

public:

Employee(int id, double sal) : employeeID(id), salary(sal) {}

};

class DisplayInfo : public Person, public Employee {

public:

DisplayInfo(const std::string& n, const std::string& addr, int id, double sal)

: Person(n, addr), Employee(id, sal) {}

void display() {

std::cout << "Employee Information:" << std::endl;

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

std::cout << "Address: " << address << std::endl;

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

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

}

};

int main() {

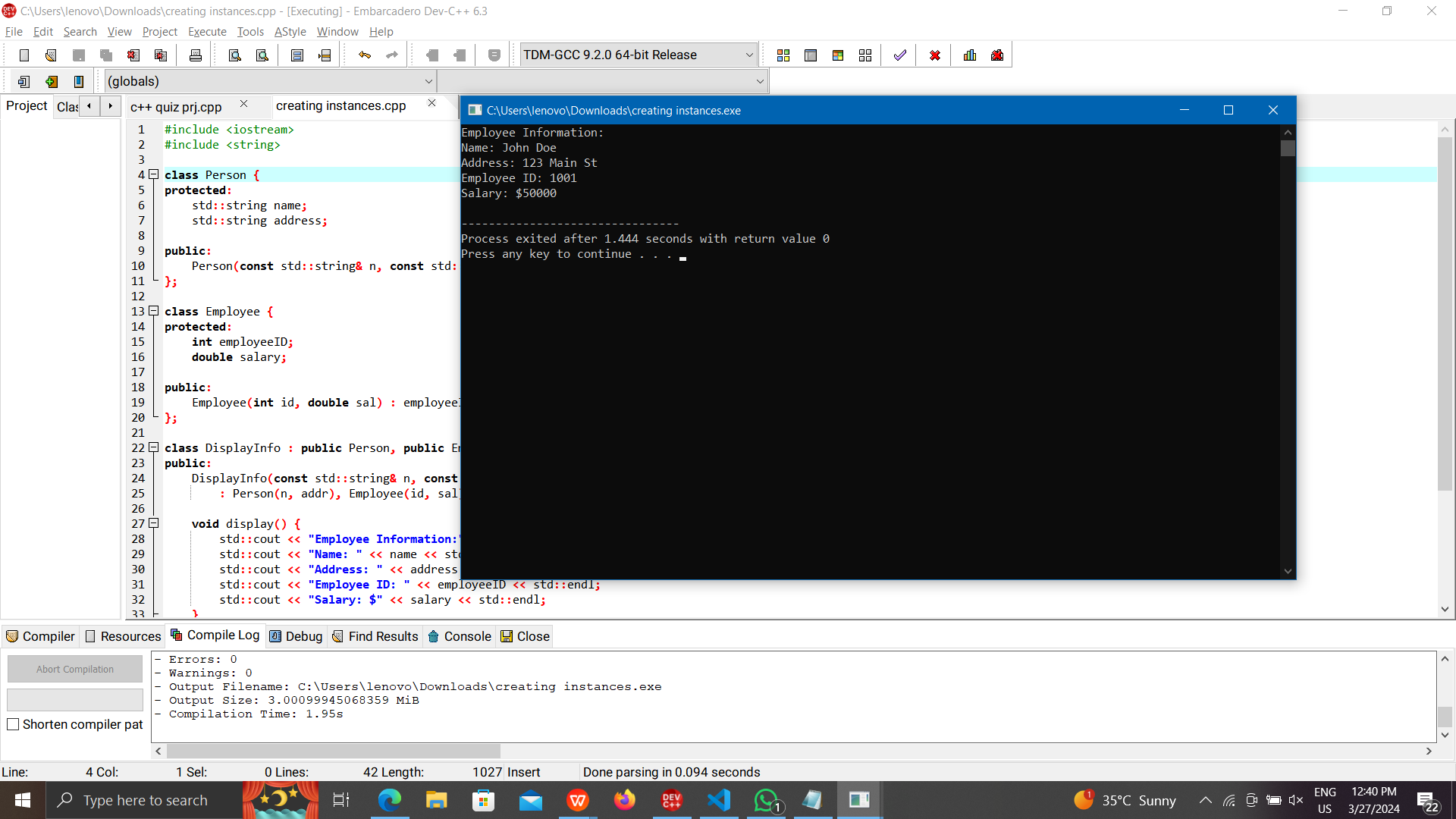
DisplayInfo emp1("John Doe", "123 Main St", 1001, 50000.0);

emp1.display();

return 0;

}

**Output:**



(3). Design a C++ program to showcase the internal usage of pointer

arithmetic by the compiler when accessing array elements.

Create a simple array of integers and utilize pointers to iterate

through the elements. The objective is to illustrate how the

compiler translates array indexing into pointer arithmetic.

**Program:**

#include <iostream>

int main() {

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

int\* ptr = arr;

std::cout << "Array elements using pointer arithmetic:" << std::endl;

for (int i = 0; i < 5; ++i) {

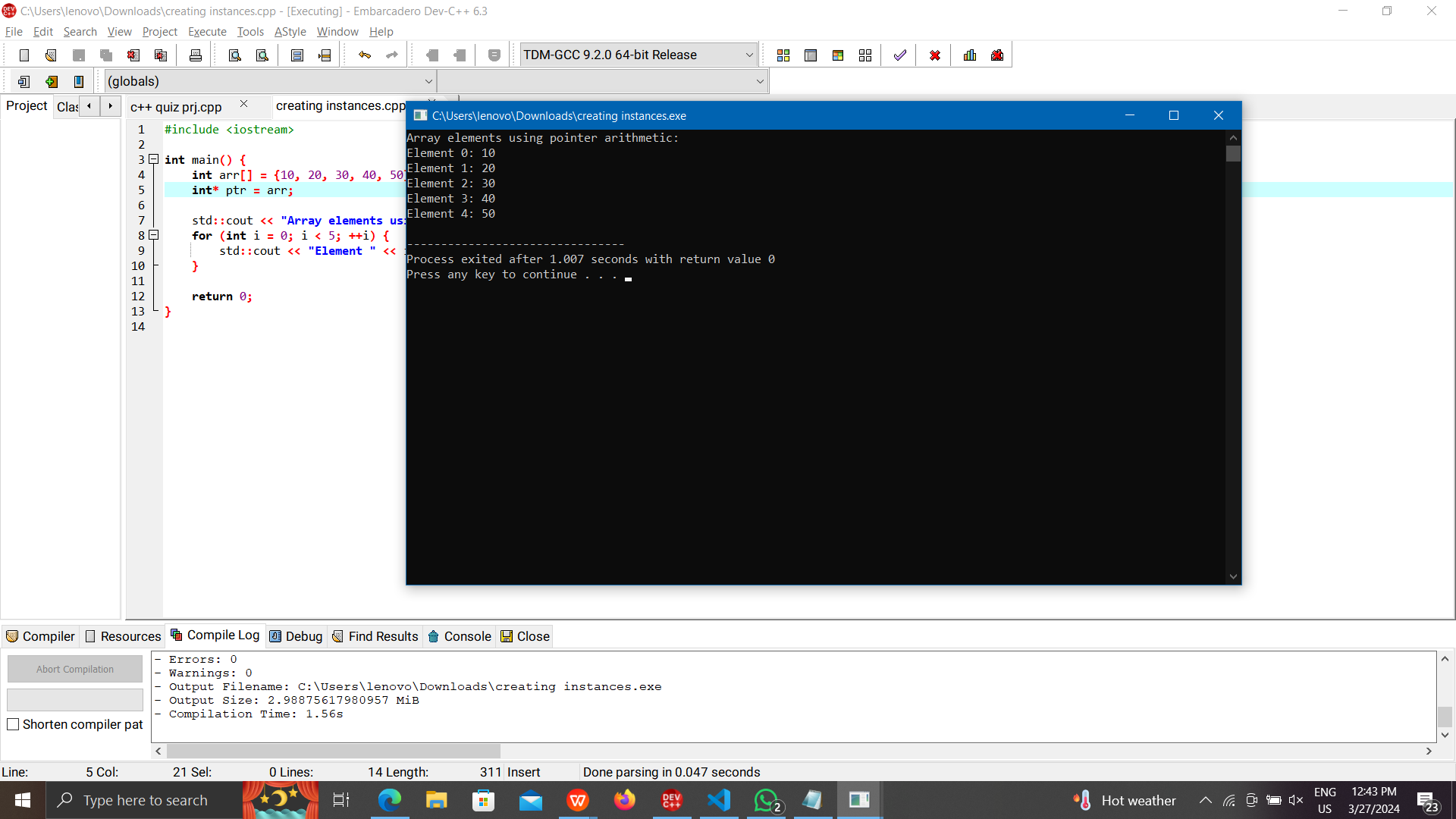
std::cout << "Element " << i << ": " << \*(ptr + i) << std::endl;

}

return 0;

}

**Output:**



(4). Develop a C++ program that illustrates the concepts of public,

private, and protected members within a class hierarchy,

showcasing their accessibility in both base and derived classes.

Create a base class named **BaseClass** with a mix of public,

private, and protected members. Then, derive a class named

**DerivedClass** from the **BaseClass** to demonstrate how these

members can be accessed through inheritance.

**Program:**

#include <iostream>

class BaseClass {

public:

int publicVar;

private:

int privateVar;

protected:

int protectedVar;

public:

BaseClass() : publicVar(1), privateVar(2), protectedVar(3) {}

void displayBase() {

std::cout << "BaseClass - Public: " << publicVar << ", Private: " << privateVar << ", Protected: " << protectedVar << std::endl;

}

};

class DerivedClass : public BaseClass {

public:

void displayDerived() {

std::cout << "DerivedClass - Public: " << publicVar << std::endl;

std::cout << "DerivedClass - Protected: " << protectedVar << std::endl;

}

};

int main() {

DerivedClass derivedObj;

derivedObj.displayBase();

derivedObj.displayDerived();

return 0;

}

**Output:**

