**OBJECT ORIENTED PROGRAMMING**

**LAB 01**

**TASK 01:**

Execute and show output of all the sample programs given in the manual.

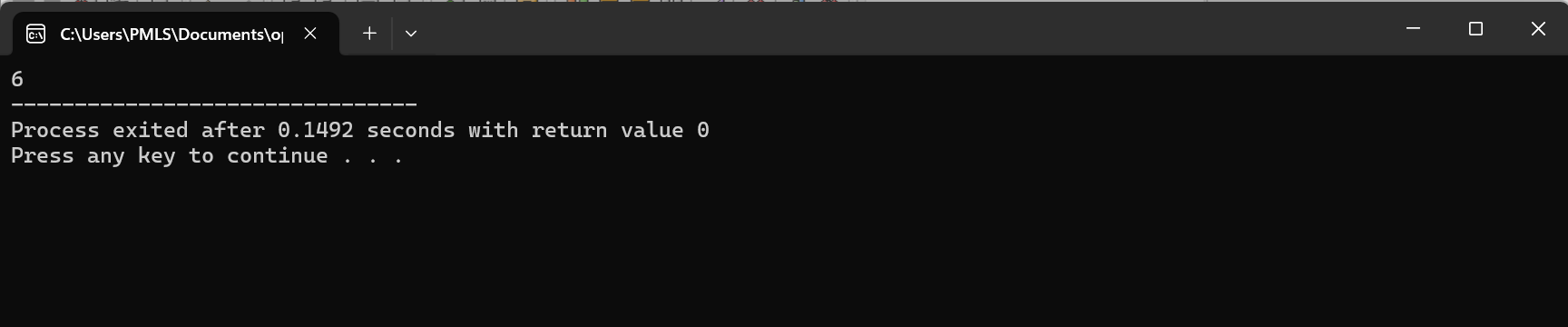
**EXAMPLE 1:**

#include <iostream>

using namespace std;

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

int main(int argc, char\*\* argv) {

int a,b;

int result;

a=5;

b=2;

a= a+1;

result= a-b;

cout << result;

return 0;}

**OUTPUT:**

**EXAMPLE 2:**

#include <iostream>

using namespace std;

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

int main(int argc, char\*\* argv) {

int a,b,c;

int result;

a=5;

b=3;

c=2;

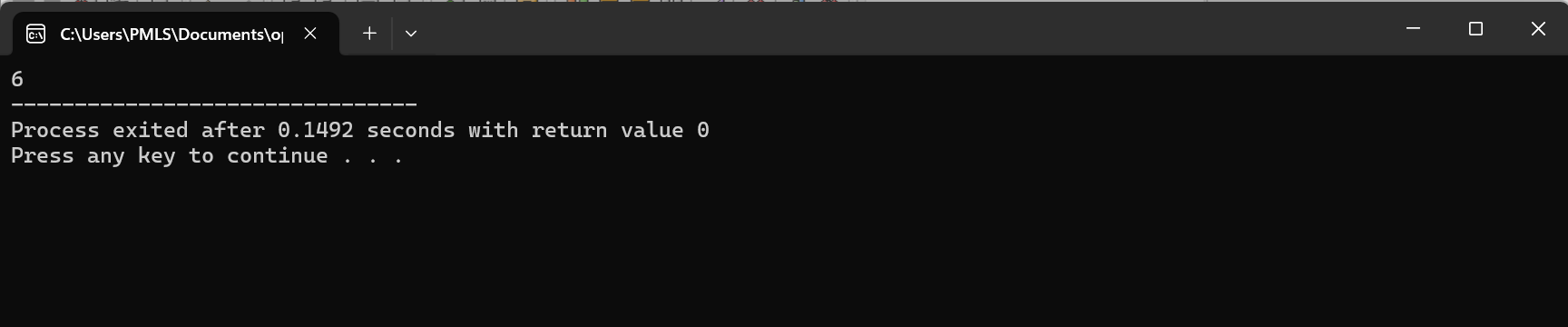
a=a+b;

result=a-c;

cout << result;

return 0;

}

**OUTPUT:**

**EXAMPLE 3:**

#include <iostream>

#include <string>

using namespace std;

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

int main(int argc, char\*\* argv) {

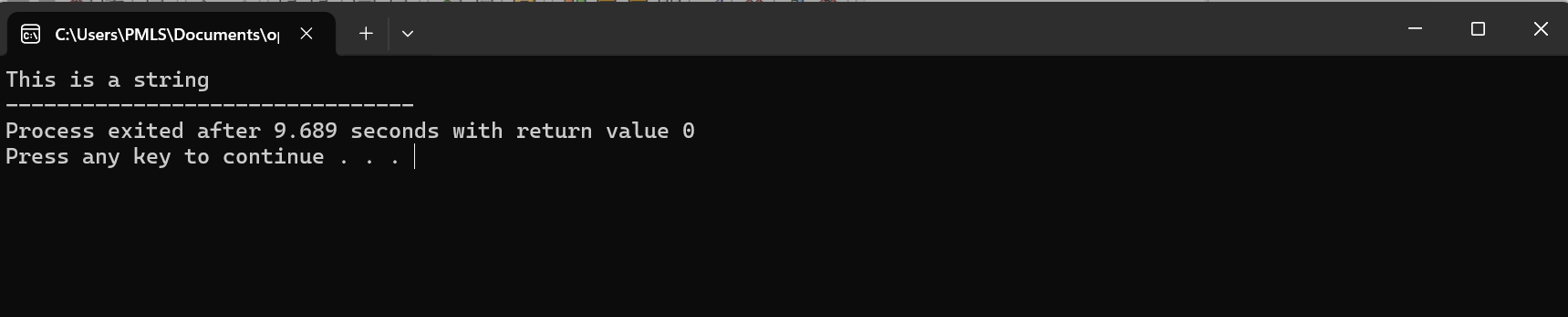
string mystring;

mystring="This is a string";

cout << mystring;

return 0;

}

**OUTPUT:**

**TASK 02:**

Task 2: Write a C++ program that will read 2 integer numbers from the user (num 1, num2) and find their summation and store it in variable (sum), then find their average and store it in variable (ave). Use the following arithmetic expression to perform the addition operation:

**CODE:**

#include <iostream>

using namespace std;

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

int main(int argc, char\*\* argv) {

int num1,num2,sum;

float avg;

cin>>num1;

cin>>num2;

sum= num1+num2;

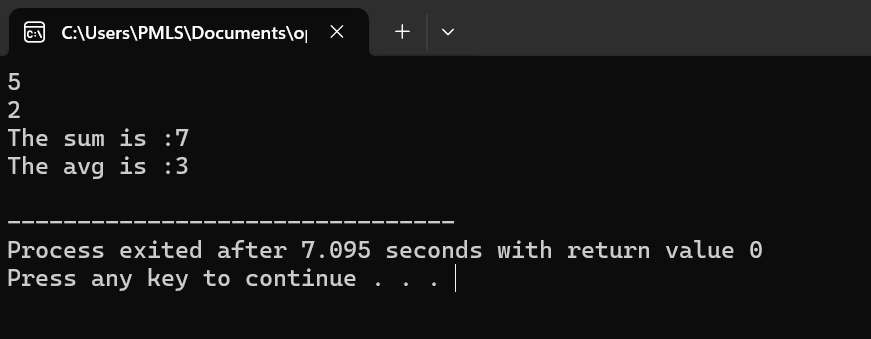
avg= static\_cast<float>(sum)/2;

cout << "The sum is :" << sum << endl;

cout << "The avg is :" << avg << endl;

return 0;}

**OUTPUT:**



**TASK 03:**

Length and Breadth of a rectangle are input through the keyboard; write a program to calculate the area and parameter of the rectangle.

Formulae: Area = Length x Breadth

Perimeter = 2 x (Length + Breadth)

**CODE:**

#include <iostream>

using namespace std;

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

int main(int argc, char\*\* argv) {

int length,Breadth;

float Area, perimeter;

cout << "The length of rectangle is :";

cin>>length;

cout << "The breadth of rectangle is :";

cin>>Breadth;

Area =length\*Breadth;

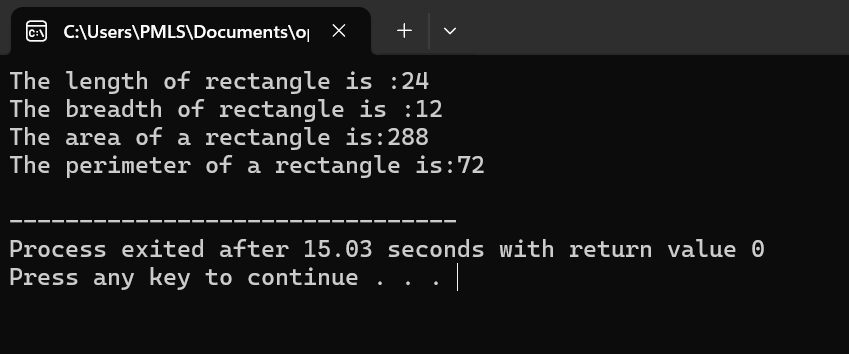
perimeter =2\*(length + Breadth);

cout << "The area of a rectangle is:" << Area <<endl;

cout << "The perimeter of a rectangle is:" << perimeter <<endl;

return 0;

}

**OUTPUT:**

**TASK 04:**

Write a program that generates the following table

1990 135

1991 7290

1992 11300

1993 16200

Use a single cout statement for all output:

**CODE:**

#include <iostream>

using namespace std;

int main() {

int years[] = {1990, 1991, 1992, 1993};

int values[] = {135, 7290, 11300, 16200};

cout << "Year\tValue\n";

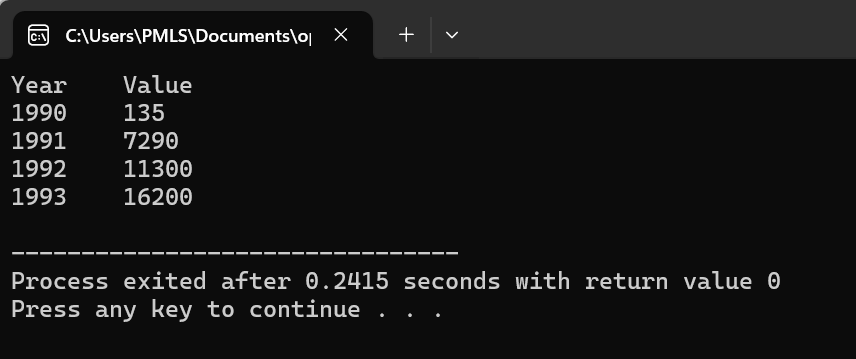
for (int i = 0; i < 4; ++i) {

cout << years[i] << "\t" << values[i] << endl;

}

return 0;

}

**OUTPUT:**

**ASSIGNMENT QUESTIONS**

**QUESTION NO 1**

Accept a character from user and display ASSCII value?

**CODE:**

#include<iostream>

#include<string>

using namespace std;

int main(){

char x;

cout << "Enter the character:";

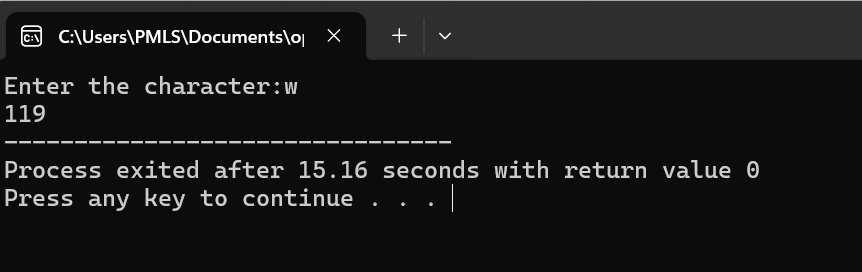
cin >> x;

int x1=x;

cout << x1;

return 0;

}

**OUTPUT:**

**QUESTION NO 02:**

Check weather the number is odd or even using ternary operator?

**CODE:**

#include<iostream>

#include<string>

using namespace std;

int main(){

int number;

cout << "ENTER A NUMBER:";

cin >> number;

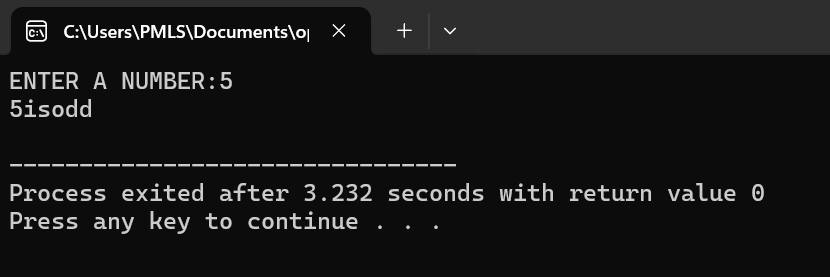
string result= (number % 2== 0)? "even" :"odd";

cout << number << " is " << result << endl;

return 0;

}

**OUTPUT:**

****

**QUESTION NO 3:**

Swap the values of two variables?

**CODE:**

#include <iostream>

using namespace std;

int main() {

int a = 5;

int b = 10;

cout << "Before swapping: a = " << a << ", b = " << b << endl;

int temp = a;

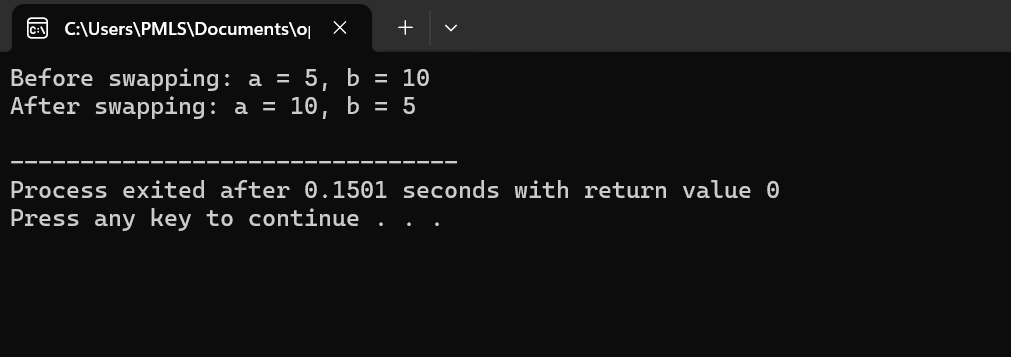
a = b;

b = temp;

cout << "After swapping: a = " << a << ", b = " << b << endl;

return 0;

}

**OUTPUT:**

**OBJECT ORIENTED PROGRAMMING**

**LAB 02**

**TASK 03:**

A digital thermostat displays the following message according to the temperature in your house, in Celsius, of your house.

**CODE:**

#include<iostream>

#include<string>

using namespace std;

int main(){

int temperature;

cout << "ENTER THE TEMPERTAURE:";

cin >> temperature;

if(temperature>=10 && temperature<=14){

cout << "cold";

}

else if(temperature>= 15 && temperature<=19){

cout << "chilly";

}

else if(temperature>= 20 && temperature<=24){

cout << "Good";

}

else if(temperature>= 25 && temperature<=29){

cout << "warm";

}

else if(temperature>=30 ){

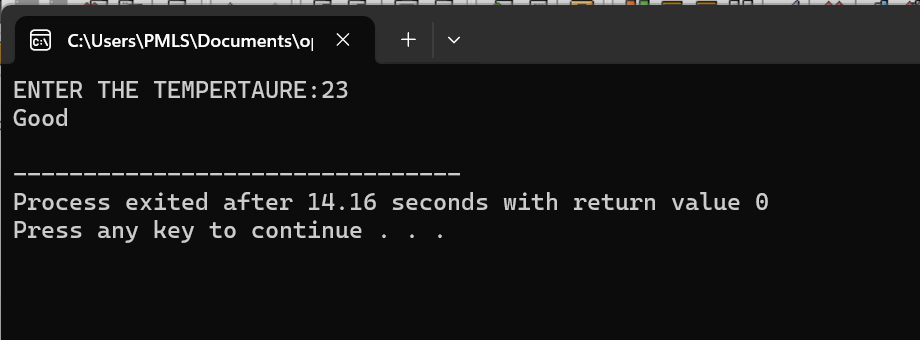
cout << "Hot";

}

cout << endl;

return 0;

}

**OUTPUT:**

**TASK 04:**

Task 4: Given a taxable income, calculate the federal tax based on the formula:

Taxable Income Tax

a) <= 29,590 17%

b)> 29,590 & <= 59,180 5,030 + 26% of the diff between T.I. and 29,590

c)>59,180 12,724 + 29% of the diff between T.I. and 59,180

**CODE:**

#include <iostream>

using namespace std;

int main() {

double taxableIncome;

double tax = 0.0;

cout << "Enter the taxable income: ";

cin >> taxableIncome;

if (taxableIncome <= 29590) {

tax = taxableIncome \* 0.17;

} else if (taxableIncome <= 59180) {

tax = 5030 + 0.26 \* (taxableIncome - 29590);

} else {

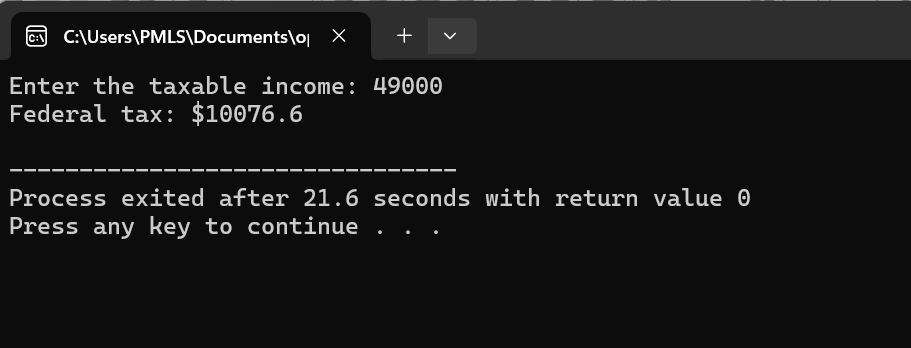
tax = 12724 + 0.29 \* (taxableIncome - 59180);

}

cout << "Federal tax: $" << tax << endl;

return 0;

}

**OUTPUT:**

**TASK 06**

use for loops to construct a program that displays a pyramid of Xs on the screen. The pyramid should look like this

X

XXX

XXXXX

XXXXXXX

XXXXXXXXX

except that it should be 20 lines high, instead of the 5 lines shown here. One way to do this is to nest two inner loops, one to print spaces and one to print Xs, inside an outer loop that steps down the screen from line to line.

**CODE:**

#include <iostream>

using namespace std;

int main() {

int height = 20;

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

for (int j = 0; j < height - i - 1; ++j) {

cout << " ";

}

for (int k = 0; k < 2 \* i + 1; ++k) {

cout << "X";

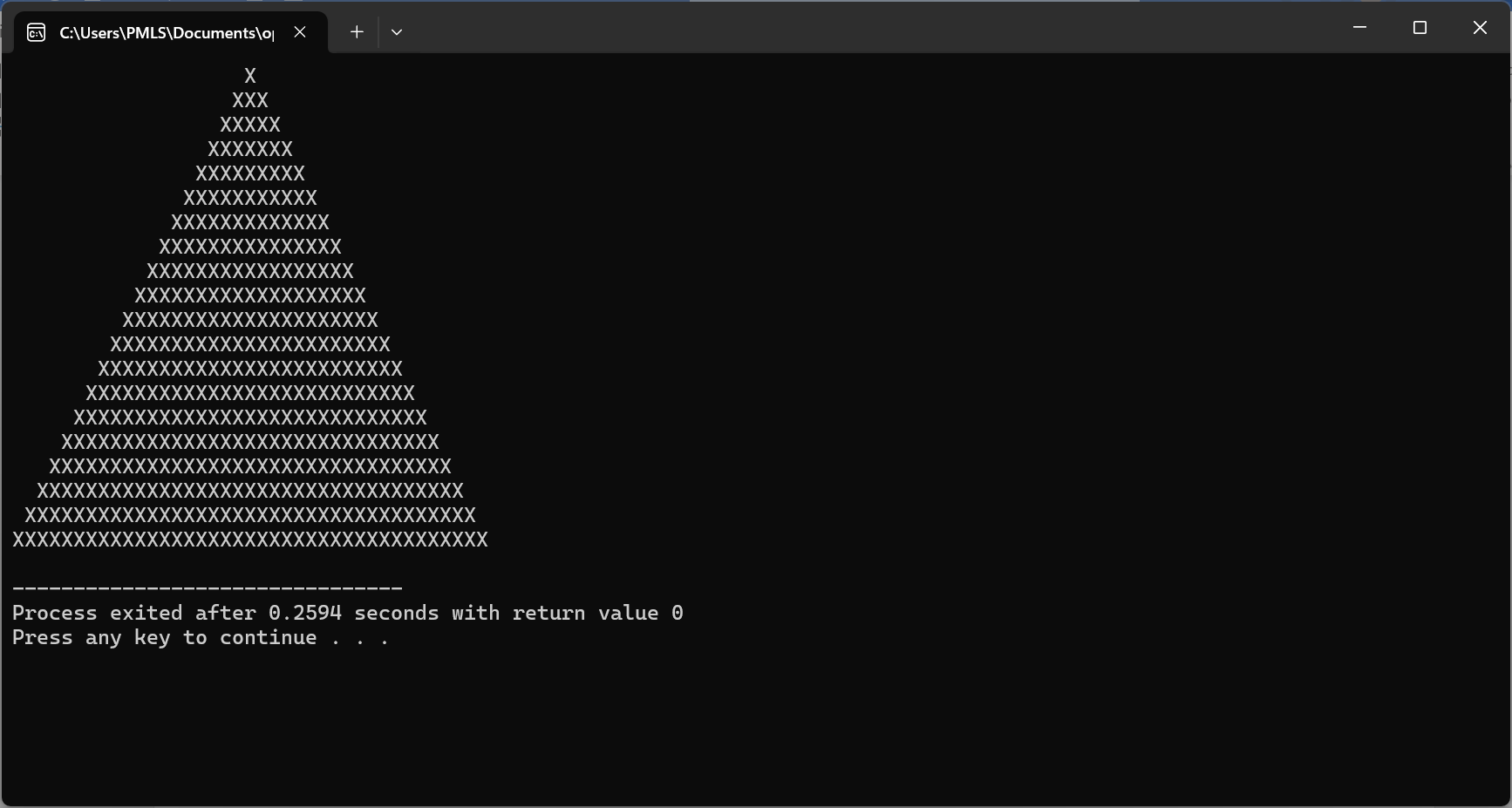
}

cout << endl;

}

return 0;

}

**OUTPUT:**

**OBJECT ORIENTED PROGRAMMING**

**LAB 03**

**Task 1:**

Create structure called employee that contains three members: an employee name (type string), employee number (type int) and employee’s compensation (in rupees; type float). Ask the user to fill in this data for four employees, store in four variables of type struct employee, and then display the information of each employee.

**CODE**

#include <iostream>

#include <string>

using namespace std;

// Structure definition for employee

struct employee {

string name;

int number;

float compensation;

};

int main() {

// Array to store information of four employees

employee employees[4];

// Input data for four employees

for (int i = 0; i < 4; ++i) {

cout << "Enter details for Employee " << i+1 << ":" << endl;

cout << "Name: ";

getline(cin, employees[i].name);

cout << "Employee Number: ";

cin >> employees[i].number;

cout << "Compensation (in rupees): ";

cin >> employees[i].compensation;

cin.ignore();

}

// Display information of each employee

cout << "\nInformation of Employees:" << endl;

for (int i = 0; i < 4; ++i) {

cout << "Employee " << i+1 << ":" << endl;

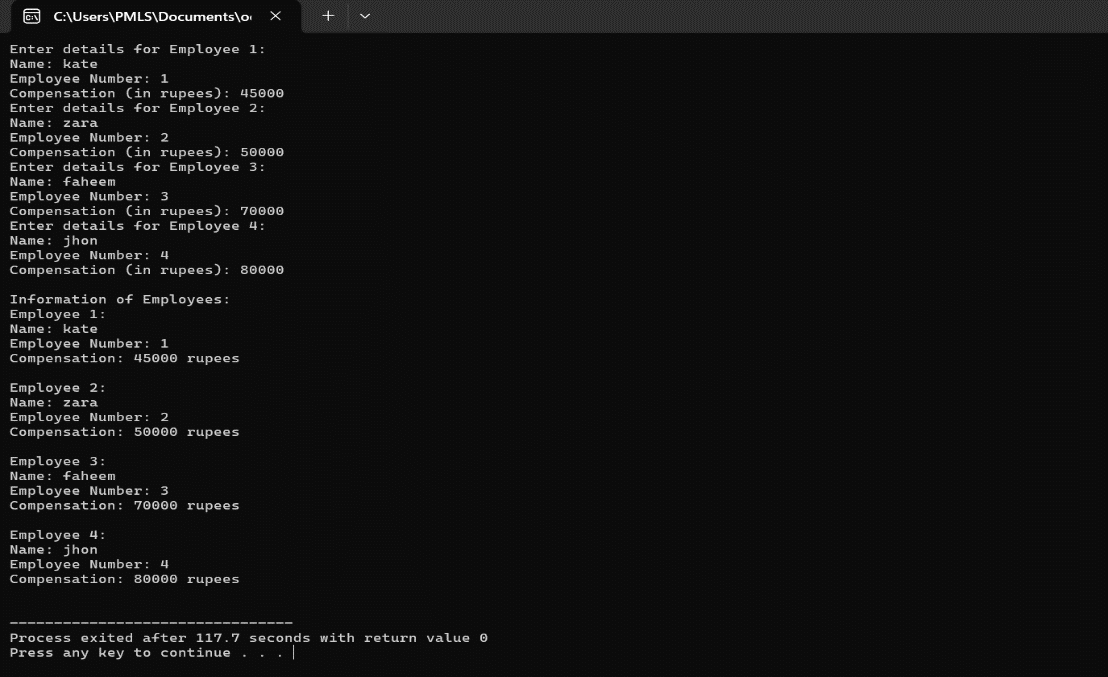
cout << "Name: " << employees[i].name << endl;

cout << "Employee Number: " << employees[i].number << endl;

cout << "Compensation: " << employees[i].compensation << " rupees" << endl;

cout << endl;

}return 0;}

**OUTPUT**

**TASK 2:**

Create a structure of type Date that contains three members: the day of month, the month, and the year, all of type int. Have the user enter a date in the format 22/06/2016, store in a variable of type struct Date, then retrieve the values from the variable and print them out in the same format.

**CODE**

#include <iostream>

#include <sstream>

#include <string>

using namespace std;

// Structure definition for Date

struct Date {

int day;

int month;

int year;

};

int main() {

Date date;

// Prompt user to enter a date

string input;

cout << "Enter a date in the format DD/MM/YYYY: ";

getline(cin, input);

// Extract day, month, and year from input

stringstream ss(input);

char discard; // For discarding '/'

ss >> date.day >> discard >> date.month >> discard >> date.year;

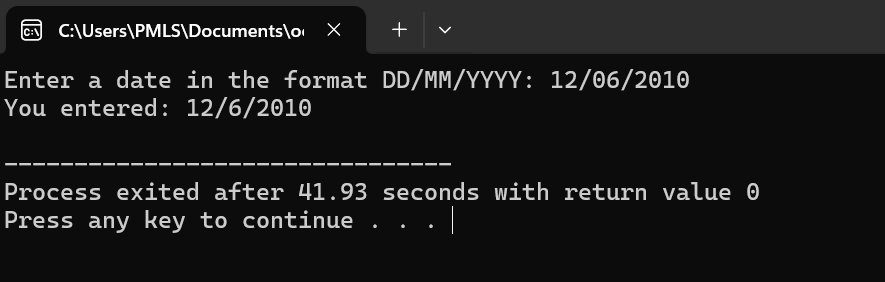
// Print out the entered date

cout << "You entered: " << date.day << "/" << date.month << "/" << date.year << endl;

return 0;

}

**OUTPUT**

****

**TASK 3:**

Write a program to create structure account that contains three members: account number, account holder’s name and balance. Open two accounts and prints initial deposit in the accounts. Deposits Rs. 1000 to xyz’s account and withdraw Rs. 500 from abc’s account and then print the balance.

**CODE**

#include <iostream>

#include <string>

using namespace std;

// Structure definition for account

struct account {

int account\_number;

string account\_holder\_name;

float balance;

};

int main() {

// Create two accounts

account abc = {1001, "abc", 0}; // Initial balance is 0

account xyz = {1002, "xyz", 0}; // Initial balance is 0

// Print initial deposit in the accounts

cout << "Initial Deposit:" << endl;

cout << abc.account\_holder\_name << "'s account (" << abc.account\_number << "): Rs. " << abc.balance << endl;

cout << xyz.account\_holder\_name << "'s account (" << xyz.account\_number << "): Rs. " << xyz.balance << endl;

// Deposit Rs. 1000 to xyz's account

xyz.balance += 1000;

// Withdraw Rs. 500 from abc's account

abc.balance -= 500;

// Print balance after deposit and withdrawal

cout << "\nBalance after transactions:" << endl;

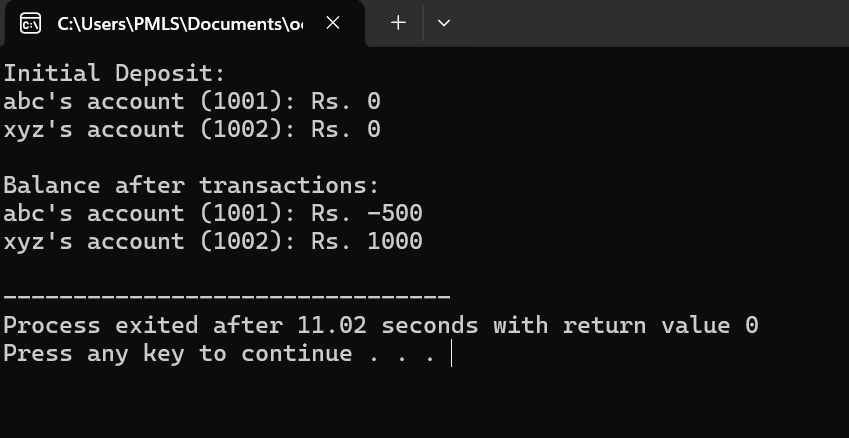
cout << abc.account\_holder\_name << "'s account (" << abc.account\_number << "): Rs. " << abc.balance << endl;

cout << xyz.account\_holder\_name << "'s account (" << xyz.account\_number << "): Rs. " << xyz.balance << endl;

return 0;

}

**OUTPUT**



**TASK 4:**

Modify the FACTOR program in this chapter so that it repeatedly asks for a number and calculates its factorial, until the user enters 0, at which point it terminates. You can enclose the relevant statements in FACTOR in a while loop or a do loop to achieve this effect.

**CODE**

#include <iostream>

using namespace std;

// Function to calculate factorial

unsigned long long factorial(int n) {

if (n == 0 || n == 1)

return 1;

else

return n \* factorial(n - 1);

}

int main() {

int number;

do {

// Ask the user to enter a number

cout << "Enter a number (enter 0 to exit): ";

cin >> number;

if (number == 0) {

cout << "Exiting..." << endl;

break; // Exit the loop if the user enters 0

}

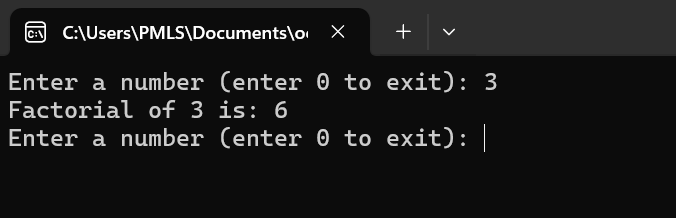
// Calculate and display factorial

cout << "Factorial of " << number << " is: " << factorial(number) << endl;

} while (true); // Loop until user enters 0

return 0;

}

**OUTPUT**

**OBJECT ORIENTED PROGRAMMING**

**LAB 04**

**TASK 1:**

1. Attempt the example on page 163 of text book.

Our first example demonstrates a simple function whose purpose is to print a line of 45 aster isks. The example program generates a table, and lines of asterisks are used to make the table more readable. Here’s the listing for TABLE:

**CODE**

#include <iostream>

using namespace std;

void starline(); //function declaration

// (prototype)

int main()

{

starline(); //call to function

cout << "Data type Range"<< endl;

starline(); //call to function

cout << "char -128 to 127"<< endl;

cout<< "short -32,768 to 32,767" << endl;

cout<< "int System dependent" << endl;

cout<< "long -2,147,483,648 to 2,147,483,647" << endl;

starline(); //call to function

return 0;

}

//-------------------------------------------------------------

// starline()

// function definition

void starline() //function declarator

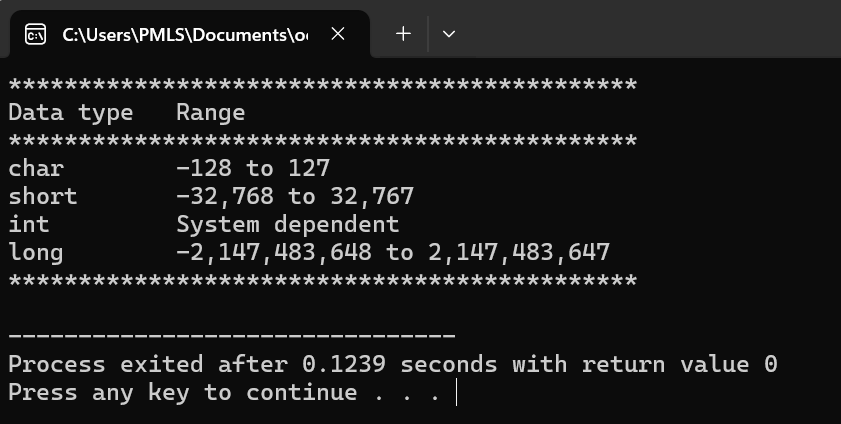
{

for(int j=0; j<45; j++) //function body

cout << "\*";

cout << endl;

}

**OUTPUT**

**TASK 2:**

Write a function that finds the area of a circle. It should take an argument of type float and return an argument of the same type. Write a main function that gets a radius value from user, call the function and display result. Area = PI \* radius \* radius PI= 3.14159F

**CODE**

#include <iostream>

using namespace std;

// Function to calculate the area of a circle

float calculateAreaOfCircle(float radius) {

const float PI = 3.14159F;

return PI \* radius \* radius;

}

int main() {

float radius;

// Prompt user for the radius value

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

cin >> radius;

// Calculate the area of the circle using the function

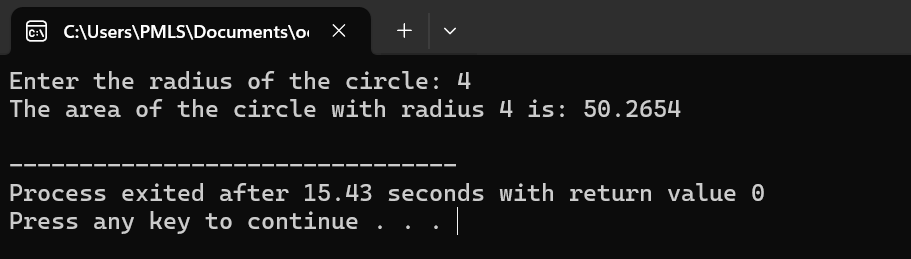
float area = calculateAreaOfCircle(radius);

// Display the result

cout << "The area of the circle with radius " << radius << " is: " << area << endl;

return 0;

}

**OUTPUT**

**TASK 3:**

Write a function called time\_to\_secs() that takes three int values for hour, minutes, and seconds as arguments, and returns the equivalent time in seconds (type long). Create a program that exercises this function by repeatedly obtaining a time value in hours, minutes, and seconds from user (format 12:59:59), calling the function, and displaying the value of seconds in returns.

**CODE**

#include <iostream>

#include <sstream>

using namespace std;

// Function to convert time to seconds

long time\_to\_secs(int hour, int minutes, int seconds) {

return (hour \* 3600) + (minutes \* 60) + seconds;

}

int main() {

int hour, minutes, seconds;

char colon; // To store the colon character

while (true) {

// Prompt the user to enter a time value

cout << "Enter the time in the format HH:MM:SS (enter 0 to exit): ";

cin >> hour >> colon >> minutes >> colon >> seconds;

// Check if the user wants to exit

if (hour == 0 && minutes == 0 && seconds == 0) {

cout << "Exiting..." << endl;

break;

}

// Call the function to convert time to seconds

long total\_seconds = time\_to\_secs(hour, minutes, seconds);

// Display the equivalent time in seconds

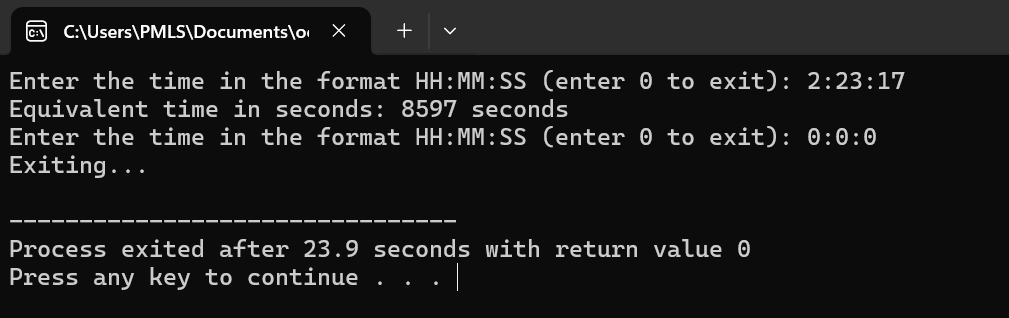
cout << "Equivalent time in seconds: " << total\_seconds << " seconds" << endl;

}

return 0;

}

**OUTPUT**

****

**TASK 4:**

Write two functions namely add ( ) with different arguments passed in each function (add two double variables, and add two integer variables in each function). Calling these functions in main () program and displaying the result of each function separately.

**CODE**

#include <iostream>

using namespace std;

// Function to add two double variables

double add(double num1, double num2) {

return num1 + num2;

}

// Function to add two integer variables

int add(int num1, int num2) {

return num1 + num2;

}

int main() {

double doubleNum1 = 5.5, doubleNum2 = 3.3;

int intNum1 = 10, intNum2 = 20;

// Call the add() function for double variables

double result1 = add(doubleNum1, doubleNum2);

cout << "Result of adding " << doubleNum1 << " and " << doubleNum2 << " (double): " << result1 << endl;

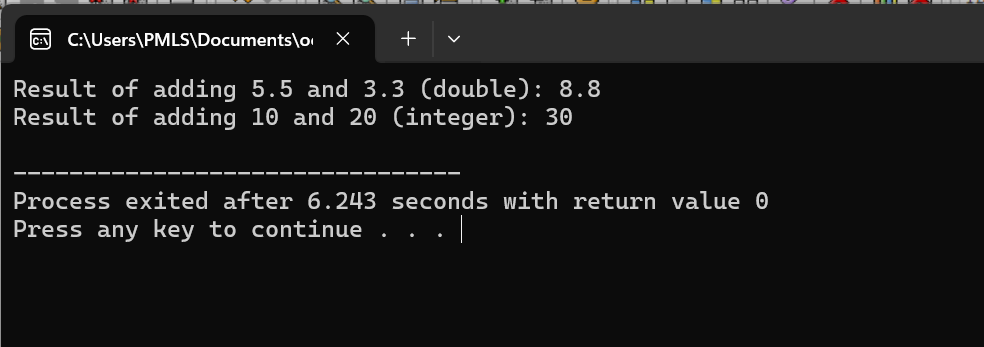
// Call the add() function for integer variables

int result2 = add(intNum1, intNum2);

cout << "Result of adding " << intNum1 << " and " << intNum2 << " (integer): " << result2 << endl;

return 0;

}

**OUTPUT**

**TASK 5:**

Is it possible to exempt the function prototype? Re-attempt the example 1 without prototype in the beginning.

**CODE**

#include <iostream>

using namespace std;

// Function to add two double variables

double add(double num1, double num2) {

return num1 + num2;

}

// Function to add two integer variables

int add(int num1, int num2) {

return num1 + num2;

}

int main() {

double doubleNum1 = 5.5, doubleNum2 = 3.3;

int intNum1 = 10, intNum2 = 20;

// Call the add() function for double variables

double result1 = add(doubleNum1, doubleNum2);

cout << "Result of adding " << doubleNum1 << " and " << doubleNum2 << " (double): " << result1 << endl;

// Call the add() function for integer variables

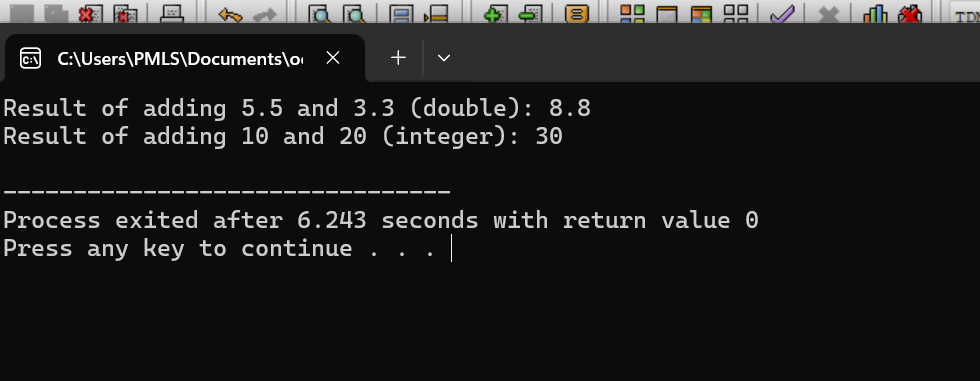
int result2 = add(intNum1, intNum2);

cout << "Result of adding " << intNum1 << " and " << intNum2 << " (integer): " << result2 << endl;

return 0;

}

**OUTPUT**



**TASK 6:**

Reform example 1 such that the character to be printed and the number of times you want it to be printed is given as input from the user.

**CODE**

#include <iostream>

using namespace std;

// Function declaration (prototype) for starline()

void printCharacter(char ch, int count);

int main() {

char character;

int count;

// Prompt the user to enter the character and count

cout << "Enter the character you want to print: ";

cin >> character;

cout << "Enter the number of times you want to print it: ";

cin >> count;

// Call to function printCharacter()

printCharacter(character, count);

return 0;

}

// Function definition for printCharacter()

void printCharacter(char ch, int count) {

// Function body - prints the character 'count' number of times

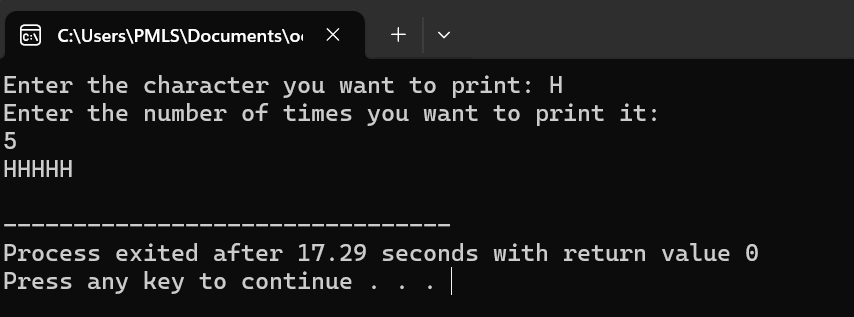
for (int i = 0; i < count; i++)

cout << ch;

cout << endl;

}

**OUTPUT**



**TASK 7:**

Attempt the example on page 216 of text book.

Recall from Chapter 1 that an object has the same relationship to a class that a variable has to a data type. An object is said to be an instance of a class, in the same way my 1954 Chevrolet is an instance of a vehicle. In SMALLOBJ, the class—whose name is smallobj—is defined in the first part of the program. Later, in main(), we define two objects—s1 and s2—that are instances of that class.

Each of the two objects is given a value, and each displays its value.

Here’s the output of the program:

Data is 1066 object s1 displayed this

Data is 1776 object s2 displayed this

**CODE**

#include <iostream>

using namespace std;

// Definition of the class smallobj

class smallobj {

private:

int data; // Private member variable

public:

// Public member function to set data

void setData(int d) {

data = d;

}

// Public member function to display data

void displayData() {

cout << "Data is " << data << endl;

}

};

int main() {

// Define two objects (instances) of class smallobj

smallobj s1, s2;

// Set data for object s1 and display it

s1.setData(1066);

cout << "object s1 displayed this------>";

s1.displayData();

// Set data for object s2 and display it

s2.setData(1776);

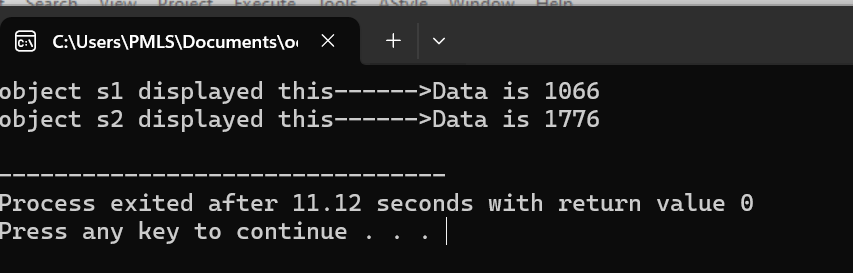
cout << "object s2 displayed this------>";

s2.displayData();

return 0;

}

**OUTPUT**

****

**TASK 8:**

Create a class called Fruit. Instantiate this class in main (create objects, 2), assign values against each variable and print the values.

**CODE**

#include <iostream>

#include <string>

using namespace std;

// Definition of the class Fruit

class Fruit {

private:

string color;

double price;

int quantity;

public:

// Constructor to initialize member variables

Fruit(string col, double pri, int quant) {

color = col;

price = pri;

quantity = quant;

}

// Function to display the details of the fruit

void displayDetails() {

cout << "Color: " << color << endl;

cout << "Price: " << price << endl;

cout << "Quantity: " << quantity << endl;

}

};

int main() {

// Instantiate two objects of class Fruit

Fruit fruit1("Red", 2.5, 10);

Fruit fruit2("Yellow", 1.8, 15);

// Print details of fruit1

cout << "Details of fruit1:" << endl;

fruit1.displayDetails();

cout << endl;

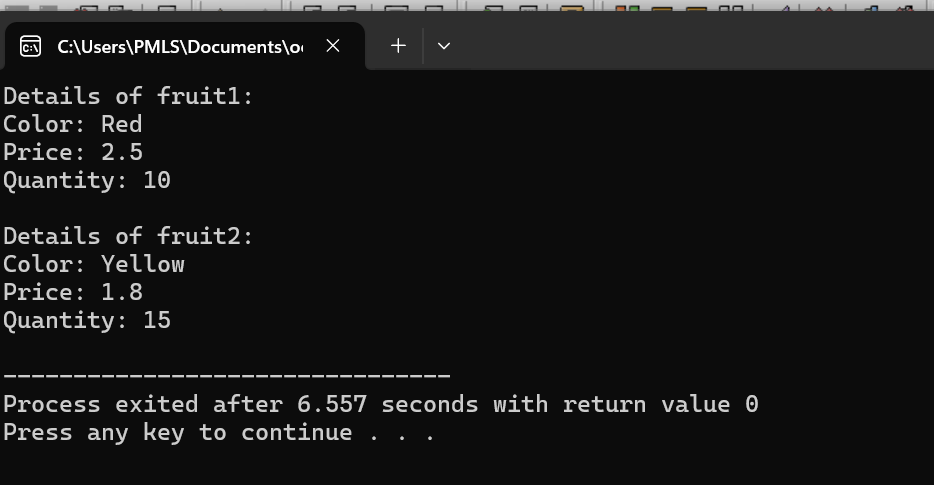
// Print details of fruit2

cout << "Details of fruit2:" << endl;

fruit2.displayDetails();

return 0;

}

**OUTPUT**

**TASK 9**

. Attempt the example on page 226 of text book.

Here’s another kind of entity C++ objects can represent: variables of a user-defined data type. We’ll use objects to represent distances measured in the English system, as discussed in Chapter 4. Here’s the listing for ENGLOBJ:

**CODE**

#include <iostream>

using namespace std;

// Definition of the Distance class

class Distance {

private:

int feet;

float inches;

public:

// Member function to set distance to given values

void setdist(int ft, float in) {

feet = ft;

inches = in;

}

// Member function to get length from user

void getdist() {

cout << "Enter feet: ";

cin >> feet;

cout << "Enter inches: ";

cin >> inches;

}

// Member function to display distance

void showdist() {

cout << feet << " feet " << inches << " inches";

}

};

int main() {

Distance dist1, dist2; // Define two distances

dist1.setdist(11, 6.25); // Set dist1

dist2.getdist(); // Get dist2 from user

// Display distances

cout << "dist1 = ";

dist1.showdist();

cout << endl;

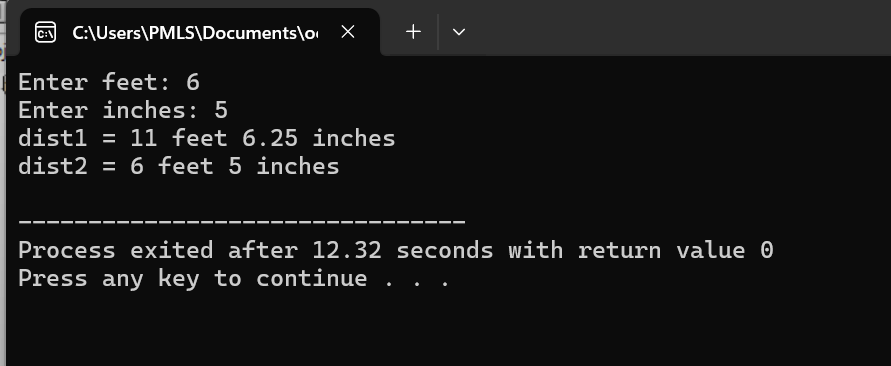
cout << "dist2 = ";

dist2.showdist();

cout << endl;

return 0;

}

**OUTPUT**

**TASK 10:**

Create an employee class. The member data should comprise an int for storing the employee number and a float for storing the employee’s compensation. Member function should allow the user to enter this data and display it. Write a main () that allow the user to enter data for three employees. And display it.

**CODE**

#include <iostream>

using namespace std;

// Employee class definition

class Employee {

private:

int employeeNumber;

float compensation;

public:

// Member function to set employee data

void setData(int empNum, float comp) {

employeeNumber = empNum;

compensation = comp;

}

// Member function to get employee data from user

void getData() {

cout << "Enter employee number: ";

cin >> employeeNumber;

cout << "Enter employee compensation: ";

cin >> compensation;

}

// Member function to display employee data

void displayData() {

cout << "Employee number: " << employeeNumber << endl;

cout << "Employee compensation: " << compensation << endl;

}

};

int main() {

Employee employees[3]; // Array to store data for three employees

// Loop to get data for each employee

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

cout << "Enter details for Employee " << i+1 << ":" << endl;

employees[i].getData(); // Get employee data from user

}

// Display data for each employee

cout << "\nEmployee Information:" << endl;

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

cout << "Employee " << i+1 << ":" << endl;

employees[i].displayData(); // Display employee data

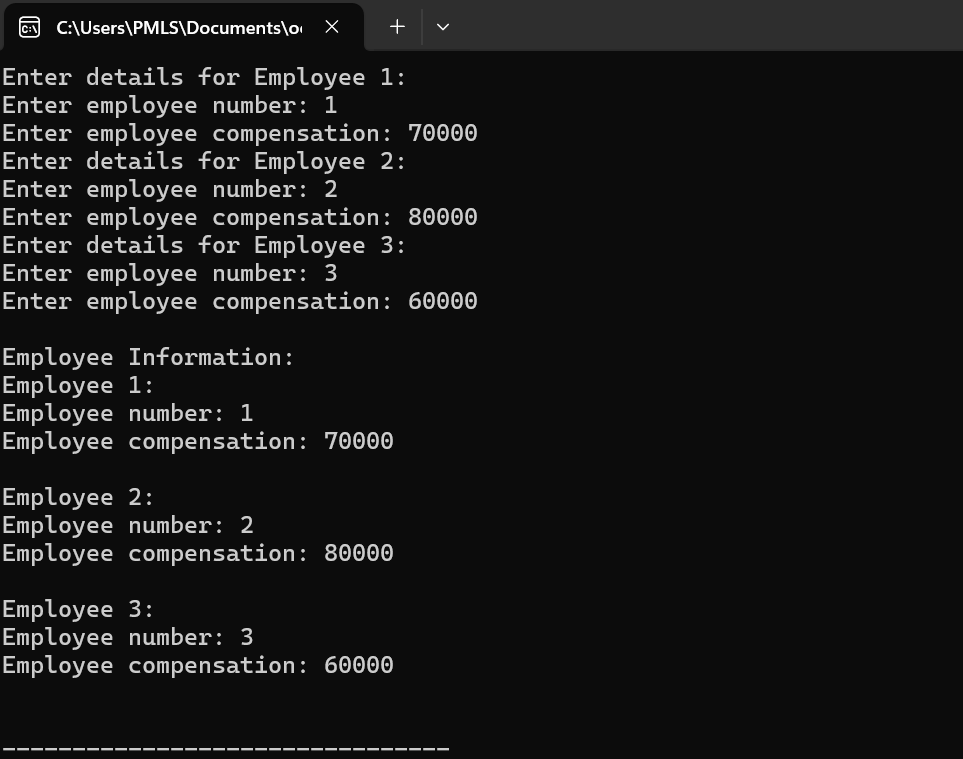
cout << endl;

}

return 0;

}

**OUTPUT**



**OBJECT ORIENTED PROGRAMMING**

**LAB 05**

**TASK 1:**

Run example 01 and 02, attach output. Moreover add a parameterized Constructor to the class Fruit

Create two instance of this class and use the constructor for initialization.

**CODE**

**Example1**

#include <iostream>

using namespace std;

class Myclass{

public:

int x;

Myclass():x(10)

{cout<<"I' am a constructor\n";

}

};

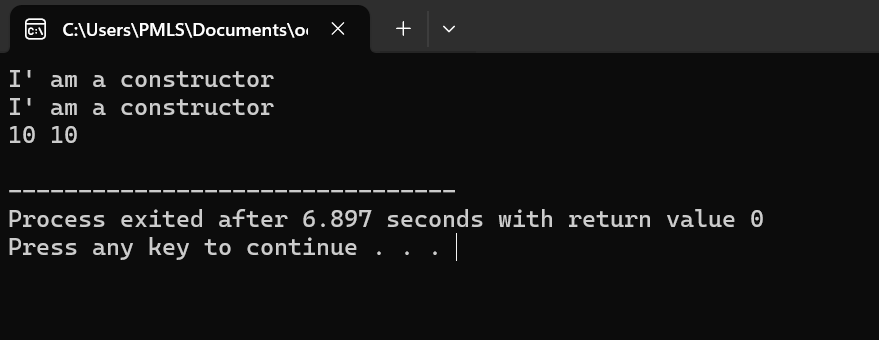
int main(){

Myclass t1,t2;

cout<<t1.x<<" "<<t2.x<<endl;

return 0;

}

**OUTPUT**

**Example 2**

#include <iostream>

using namespace std;

class Myclass{

public:

int x;

{ }

};

int main(){

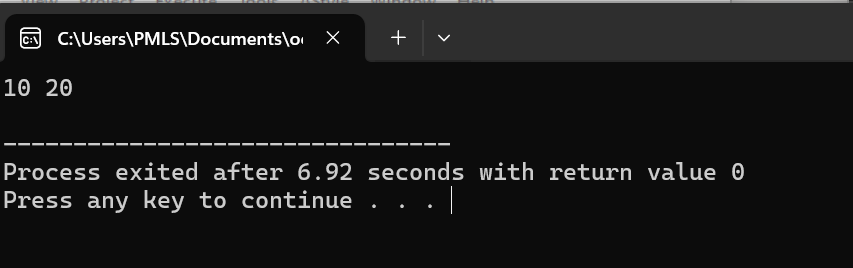
Myclass t1(10),t2(20);

cout<<t1.x<<" "<<t2.x<<endl;

return 0;

}

**OUTPUT**



**Class of fruits**

#include <iostream>

#include <string>

using namespace std;

class Fruit {

private:

string color;

float price;

int quantity;

public:

// Parameterized constructor

Fruit(string c, float p, int q) {

color = c;

price = p;

quantity = q;

}

// Function to print fruit details

void printDetails() {

cout << "Color: " << color << endl;

cout << "Price: " << price << endl;

cout << "Quantity: " << quantity << endl;

}

};

int main() {

// Creating two instances of the Fruit class using parameterized constructor

Fruit fruit1("Red", 2.5, 10);

Fruit fruit2("Yellow", 1.8, 15);

// Printing details of fruit1

cout << "Details of fruit1:" << endl;

fruit1.printDetails();

cout << endl;

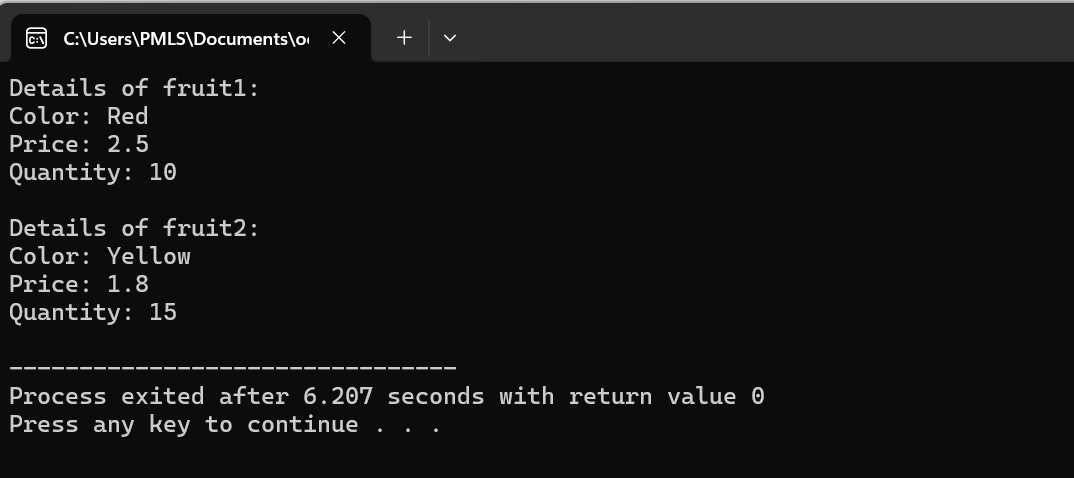
// Printing details of fruit2

cout << "Details of fruit2:" << endl;

fruit2.printDetails();

return 0;

}

**OUTPUT**

**TASK 4:**

Create a class BankAccount whose data members are private integer account number and double balance. BankAccount constructor initializes each data member. Class takes two member functions namely setdata() which takes two arguments integer and double, and showdata() to display result on screen. The member function should be defined outside the class. Create three accounts for three persons.

**CODE**

#include <iostream>

using namespace std;

class BankAccount {

private:

int accountNumber;

double balance;

public:

// Constructor to initialize accountNumber and balance

BankAccount(int accNum, double bal) {

accountNumber = accNum;

balance = bal;

}

// Function to display account data

void showData() {

cout << "Account Number: " << accountNumber << endl;

cout << "Balance: " << balance << endl;

}

};

int main() {

// Creating three accounts for three persons

BankAccount account1(1001, 5000.0);

BankAccount account2(1002, 8000.0);

BankAccount account3(1003, 10000.0);

// Displaying account data for each account

cout << "Details of Account 1:" << endl;

account1.showData();

cout << endl;

cout << "Details of Account 2:" << endl;

account2.showData();

cout << endl;

cout << "Details of Account 3:" << endl;

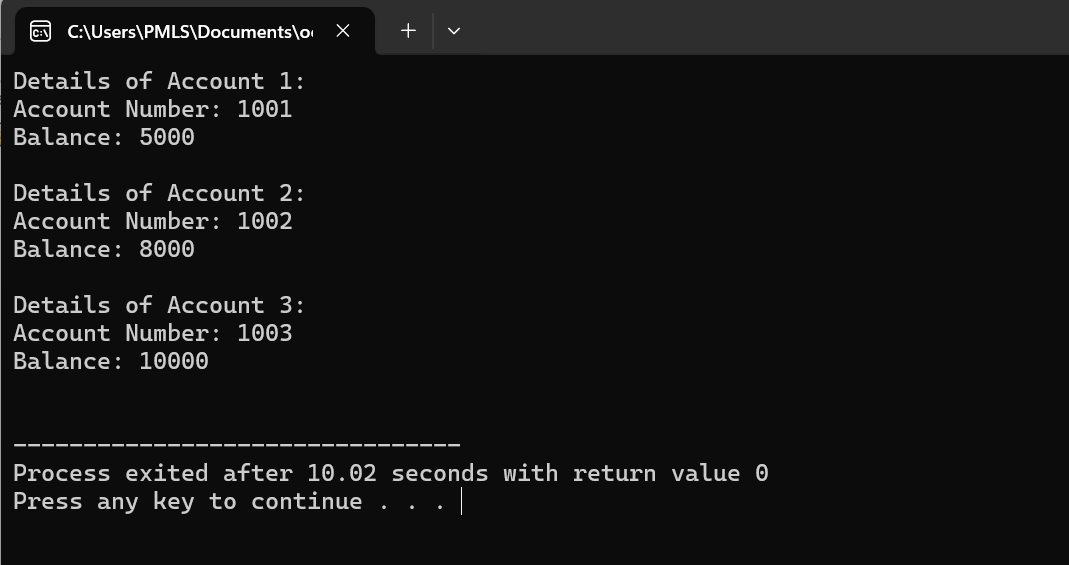
account3.showData();

cout << endl;

return 0;

}

**OUTPUT**



**TASK 6:**

Create a class called time that has separate int member data for hours, minutes, and seconds. One constructor should initialize this data to 0, and another should initialize it to fixed values. Another member function should display it, in 11:59:59 format. The final member function should add two objects of type time passed as arguments. A main() program should create two initialized time objects and one that is not initialized. Then it should add the two initialized values together, leaving the result in the third time variable. Finally it should display the value of this third variable.

**CODE**

#include <iostream>

#include <iomanip>

using namespace std;

class Time {

private:

int hours;

int minutes;

int seconds;

public:

// Constructor to initialize time data to 0

Time() : hours(0), minutes(0), seconds(0) {}

// Constructor to initialize time data to fixed values

Time(int h, int m, int s) : hours(h), minutes(m), seconds(s) {}

// Function to display time in hh:mm:ss format

void displayTime() {

cout << setfill('0') << setw(2) << hours << ":" << setw(2) << minutes << ":" << setw(2) << seconds << endl;

}

// Function to add two Time objects

Time addTime(const Time& t) {

Time sum;

sum.hours = hours + t.hours;

sum.minutes = minutes + t.minutes;

sum.seconds = seconds + t.seconds;

// Adjust minutes and seconds if they exceed their respective ranges

if (sum.seconds >= 60) {

sum.minutes++;

sum.seconds -= 60;

}

if (sum.minutes >= 60) {

sum.hours++;

sum.minutes -= 60;

}

return sum;

}

};

int main() {

// Create two initialized Time objects

Time time1(10, 30, 45);

Time time2(1, 15, 20);

// Create one Time object that is not initialized

Time time3;

// Add the two initialized values together and store the result in time3

time3 = time1.addTime(time2);

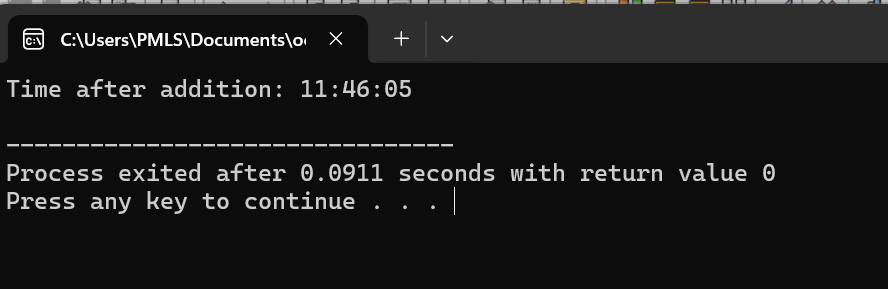
// Display the value of the third variable

cout << "Time after addition: ";

time3.displayTime();

return 0;

}

**OUTPUT**

**TASK 7:**

Execute complete example 06.

**CODE**

#include <iostream>

using namespace std;

// Definition of the Distance class

class Distance {

private:

int feet;

float inches;

public:

// Member function to set distance to given values

void setdist(int ft, float in) {

feet = ft;

inches = in;

}

// Member function to get length from user

void getdist() {

cout << "Enter feet: ";

cin >> feet;

cout << "Enter inches: ";

cin >> inches;

}

// Member function to display distance

void showdist() {

cout << feet << " feet " << inches << " inches";

}

};

int main() {

Distance dist1, dist2; // Define two distances

dist1.setdist(11, 6.25); // Set dist1

dist2.getdist(); // Get dist2 from user

// Display distances

cout << "dist1 = ";

dist1.showdist();

cout << endl;

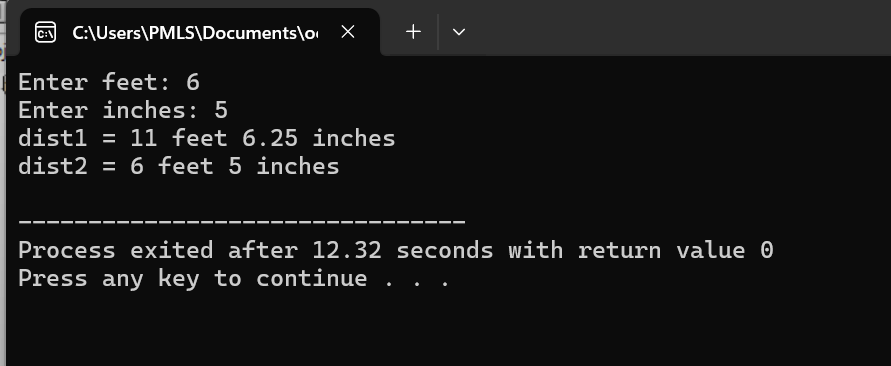
cout << "dist2 = ";

dist2.showdist();

cout << endl;

return 0;

}

**OUTPUT**