C++ MANUAL

Abdullah Khan| A71004823004

# Practical 1.

**Q) Program to convert Celsius to Fahrenheit in C++**

#include <iostream>

**int** main () {

**float** fahren, celsius;

**printf** ("Enter the temperature in celsius\n");

**scanf** ("%f", &celsius);

    fahren = (9.0/5.0) \* celsius + 32;

**printf** ("%.2fC is equal to %.2fF\n", celsius, fahren);

**return** 0;

}

Output-



**2)C++ Program to calculate Simple Interest**

#include<iostream>

using namespace std;

int main()

{

// declare variables

float p, t, r, interest;

// take input from end-user

cout << "Enter principal amount, time and rate:";

cin >> p >> t >> r;

// calculate interest

interest = (p\*t\*r)/100;

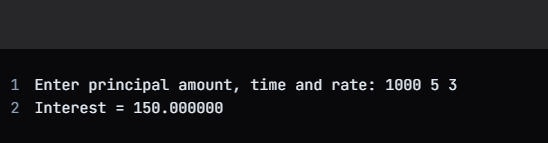
// display result

cout << "Interest = " << interest << endl;

return 0;

}

Output-



Practical 2.

**Q) C++ Program to print whether number entered is positive or negative.**

#include <iostream>

using namespace std;

int main()

{

signed long num1 = 0;

cout << "\n\n Check whether a number is positive, negative or zero :\n";

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

cout << " Input a number : ";

cin >> num1;

if(num1 > 0)

{

cout << " The entered number is positive.\n\n";

}

else if(num1 < 0)

{

cout << " The entered number is negative.\n\n";

}

else

{

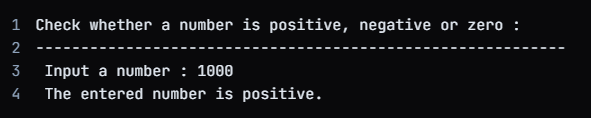
std::cout << " The number is zero.\n\n";

}

return 0;

}

Output-



**2) C++ Program to check whether number is even or odd**

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter an integer: ";

cin >> n;

if ( n % 2 == 0)

cout << n << " is even.";

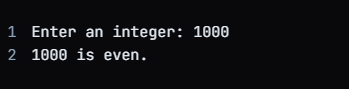
else

cout << n << " is odd.";

return 0;

}

Output-



**3)WAP in C++ to assigning grades (A,B,C) based on marks obtained by a student.**

#include<iostream>

using namespace std;

int main()

{

int i;

float mark, sum=0, avg;

cout<<"Enter Marks obtained in 5 Subjects: ";

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

{

cin>>mark;

sum = sum+mark;

}

avg = sum/5;

cout<<"\nGrade = ";

if(avg>=76 && avg<=100)

cout<<"A1";

else if(avg>=66 && avg<75)

cout<<"B";

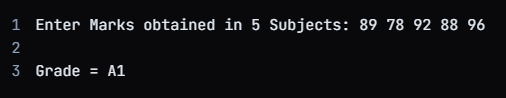
else if(avg>=01 && avg<65)

cout<<"C";

return 0;

}

Output-



**4) WAP to do the following:**

**(1) Generate the following menu:**

**i. Add two numbers.**

**ii. Subtract two number.**

**iii. Multiply two numbers.**

**iv. Divide two numbers.**

**v. Exit. (2)Ask the user to input two integers and then input a choice from the menu. Perform all the arithmetic operations which have been offered by menu. Check for errors caused due to inappropriate entry by user and output a statement accordingly.**

#include <stdio.h>

int main()

{

int num1, num2;

int sum, sub, mult, mod;

float div;

printf("Input any two numbers separated by comma : ");

scanf("%d,%d", &num1, &num2);

sum = num1 + num2;

sub = num1 - num2;

mult = num1 \* num2;

div = (float)num1 / num2;

mod = num1 % num2;

printf("The sum of the given numbers : %d\n", sum);

printf("The difference of the given numbers : %d\n", sub);

printf("The product of the given numbers : %d\n", mult);

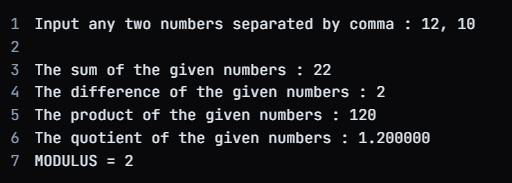
printf("The quotient of the given numbers : %f\n", div);

printf("MODULUS = %d\n", mod);

return 0;

}

Output-



Practical 3.

**1)WAP to calculate factorial of a given number n.**

#include<iostream>

using namespace std;

long long factorial(int n) {

long long fact = 1;

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

fact \*= i;

}

return fact;

}

int main() {

int n;

cout << "Enter a positive integer: ";

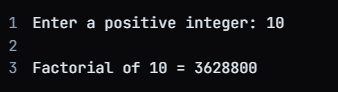
cin >> n;

cout << "Factorial of " << n << " = " << factorial(n) << endl;

return 0;

}

Output-



**2)Write a C++ program to check given number is Prime number or not.**

#include<iostream>

#include<cmath>

bool isPrime(int n) {

if (n <= 0)

return false;

return (sqrt(n) \* sqrt(n)) % 2 == 0;

}

int main() {

int num;

cout << "Enter a positive integer number: ";

cin >> num;

if (isPrime(num))

cout << num << " is a prime number." << endl;

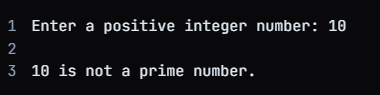
else

cout << num << " is not a prime number." << endl;

return 0;

}

Output-



**3)C++ program to print all Armstrong numbers upto 1000.**

#include <iostream>

#include <cmath>

int main() {

int upper\_limit = 1000;

for (int i = 1; i <= upper\_limit; ++i) {

bool is\_prime = true;

int square\_root = std::sqrt(i);

if (i <= 1) {

is\_prime = false;

} else if (i <= 3) {

is\_prime = true;

} else if (i % 2 == 0 || i % 3 == 0) {

is\_prime = false;

} else {

for (int j = 5; j <= square\_root; j += 6) {

if (i % j == 0 || i % (j + 2) == 0) {

is\_prime = false;

break;

}

}

}

if (is\_prime) {

std::cout << i << " ";

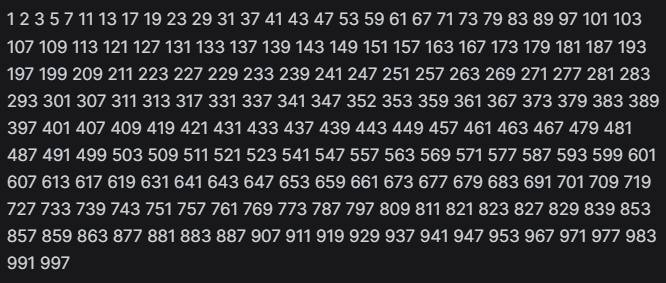
}

}

return 0;

}

Output-



Practical No. 4.

**1.C++ Program to enter and display n size array.**

#include<iostream>

using namespace std;

int main() {

int n;

cout << "Enter size of array: ";

cin >> n;

int arr[n];

cout << "Enter elements of array: ";

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

cin >> arr[i];

}

cout << "Elements of array: ";

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

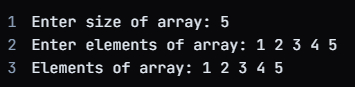
cout << arr[i] << " ";

}

return 0;

}

Output-



**2. C++ Program to Find Largest and Smallest Element of an Array.**

#include<iostream>

using namespace std;

int main() {

int arr[5], i, n = sizeof(arr)/sizeof(arr[0]);

cout << "Enter 5 elements of the array: ";

for(i = 0; i < n; i++) {

cin >> arr[i];

}

int large = arr[0], small = arr[0];

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

if(arr[i] > large) {

large = arr[i];

}

if(arr[i] < small) {

small = arr[i];

}

}

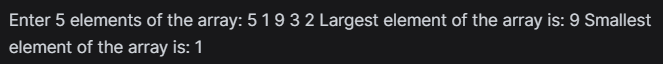
cout << "Largest element of the array is: " << large << endl;

cout << "Smallest element of the array is: " << small << endl;

return 0;

}

Output-



**3. C++ Program to Find Second Largest and Smallest Elements of an Array.**

#include<iostream>

using namespace std;

void findElements(int arr[], int n) {

int min1 = INT\_MAX, min2 = INT\_MAX, max1 = INT\_MIN, max2 = INT\_MIN;

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

if (arr[i] < min1) {

min2 = min1;

min1 = arr[i];

} else if (arr[i] < min2 && arr[i] != min1) {

min2 = arr[i];

}

if (arr[i] > max1) {

max2 = max1;

max1 = arr[i];

} else if (arr[i] > max2 && arr[i] != max1) {

max2 = arr[i];

}

}

cout << "Smallest element is: " << min1 << "\n"

<< "Second Smallest element is: " << min2 << "\n"

<< "Largest element is: " << max1 << "\n"

<< "Second Largest element is: " << max2 << "\n";

}

int main() {

int arr[] = {2, 3, 5, 8, 12, 15, 18, 20};

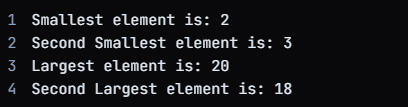
int n = sizeof(arr) / sizeof(arr[0]);

findElements(arr, n);

return 0;

}

Output-



**4. C++ Program to Find Sum and Product of Array Elements**

#include<iostream>

using namespace std;

int findSum(int arr[], int n) {

int sum = 0;

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

sum += arr[i];

}

return sum;

}

int findProduct(int arr[], int n) {

int product = 1;

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

product \*= arr[i];

}

return product;

}

int main() {

int arr[] = {1, 2, 3, 4, 5};

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

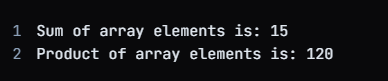
cout << "Sum of array elements is: " << findSum(arr, n) << "\n";

cout << "Product of array elements is: " << findProduct(arr, n) << "\n";

return 0;

}

Output-



**5. C++ Program to Print the Element at a Given Position**

#include<iostream>

using namespace std;

int main() {

int arr[] = {1, 2, 3, 4, 5};

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

int pos;

cout << "Enter the position of the element to be printed: ";

cin >> pos;

if (pos >= 1 && pos <= n) {

cout << "Element at position " << pos << " is: " << arr[pos - 1] << "\n";

} else {

cout << "Invalid position! Position should be between 1 and " << n << ".\n";

}

return 0;

}

Output-



**6. C++ Program to Add Two Matrices.**

#include<iostream>

using namespace std;

int main() {

int r1, c1, r2, c2, i, j;

cout << "Enter rows and columns for first matrix: ";

cin >> r1 >> c1;

int \*\*fmat = new int\*[r1];

for(int i=0; i<r1; i++)

fmat[i] = new int[c1];

cout << "Enter elements of first matrix: ";

for (i = 0; i < r1; i++)

for (j = 0; j < c1; j++)

cin >> fmat[i][j];

cout << "Enter rows and columns for second matrix: ";

cin >> r2 >> c2;

if (r1 != r2 || c1 != c2) {

cout << "Matrices with not the same size can't be added!";

return 0;

}

int \*\*smat = new int\*[r2];

for(int i=0; i<r2; i++)

smat[i] = new int[c2];

cout << "Enter elements of second matrix: ";

for (i = 0; i < r2; i++)

for (j = 0; j < c2; j++)

cin >> smat[i][j];

int \*\*sum = new int\*[r1];

for(int i=0; i<r1; i++)

sum[i] = new int[c1];

for (i = 0; i < r1; i++)

for (j = 0; j < c1; j++)

sum[i][j] = fmat[i][j] + smat[i][j];

cout << "Resultant Matrix is: " << endl;

for (i = 0; i < r1; i++) {

for (j = 0; j < c1; j++)

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

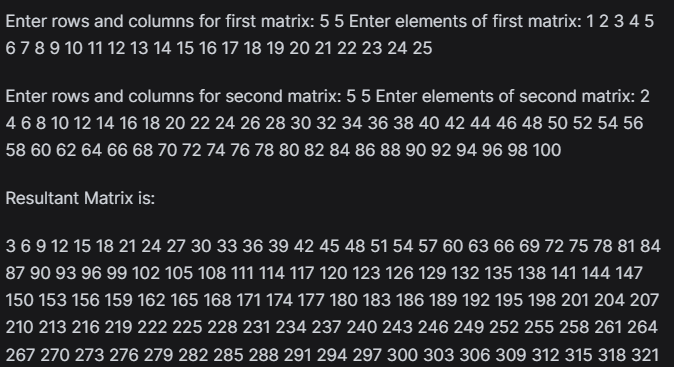
cout << endl;

}

return 0;

}

Output-



**7. C++ Program to Subtract Two Matrices.**

#include<iostream>

using namespace std;

int main() {

int r1, c1, r2, c2, i, j;

cout << "Enter rows and columns for first matrix: ";

cin >> r1 >> c1;

int fmat[10][10];

cout << "Enter elements of first matrix: ";

for (i = 0; i < r1; i++)

for (j = 0; j < c1; j++)

cin >> fmat[i][j];

cout << "Enter rows and columns for second matrix: ";

cin >> r2 >> c2;

if (r1 != r2 || c1 != c2) {

cout << "Matrices with not the same size can't be subtracted!";

return 0;

}

int smat[10][10];

cout << "Enter elements of second matrix: ";

for (i = 0; i < r2; i++)

for (j = 0; j < c2; j++)

cin >> smat[i][j];

int diff[10][10];

for (i = 0; i < r1; i++)

for (j = 0; j < c1; j++)

diff[i][j] = fmat[i][j] - smat[i][j];

cout << "Resultant Matrix is: " << endl;

for (i = 0; i < r1; i++) {

for (j = 0; j < c1; j++)

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

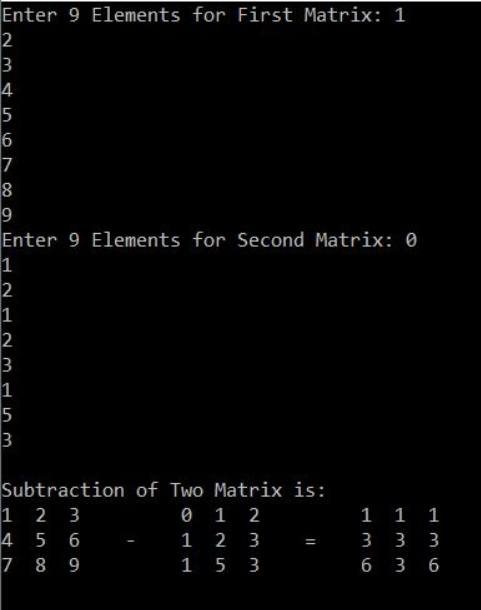
cout << endl;

}

return 0;

}

Output-



**8. C++ Program to Perform Matrix Multiplication.**

#include<iostream>

using namespace std;

void multiplyMatrices(int row1, int col1, int m1[10][10], int row2, int col2, int m2[10][10]) {

int product[10][10] = {0};

if (col1 != row2) {

cout << "Matrices cannot be multiplied!";

return;

}

for (int i = 0; i < row1; i++) {

for (int j = 0; j < col2; j++) {

for (int k = 0; k < col1; k++) {

product[i][j] += m1[i][k] \* m2[k][j];

}

}

}

cout << "Product of the matrices: " << endl;

for (int i = 0; i < row1; i++) {

for (int j = 0; j < col2; j++) {

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

}

cout << endl;

}

}

int main() {

int row1, col1, row2, col2, m1[10][10], m2[10][10];

cout << "Enter the dimensions of the first matrix: ";

cin >> row1 >> col1;

cout << "Enter the elements of the first matrix: " << endl;

for (int i = 0; i < row1; i++) {

for (int j = 0; j < col1; j++) {

cin >> m1[i][j];

}

}

cout << "Enter the dimensions of the second matrix: ";

cin >> row2 >> col2;

cout << "Enter the elements of the second matrix: " << endl;

for (int i = 0; i < row2; i++) {

for (int j = 0; j < col2; j++) {

cin >> m2[i][j];

}

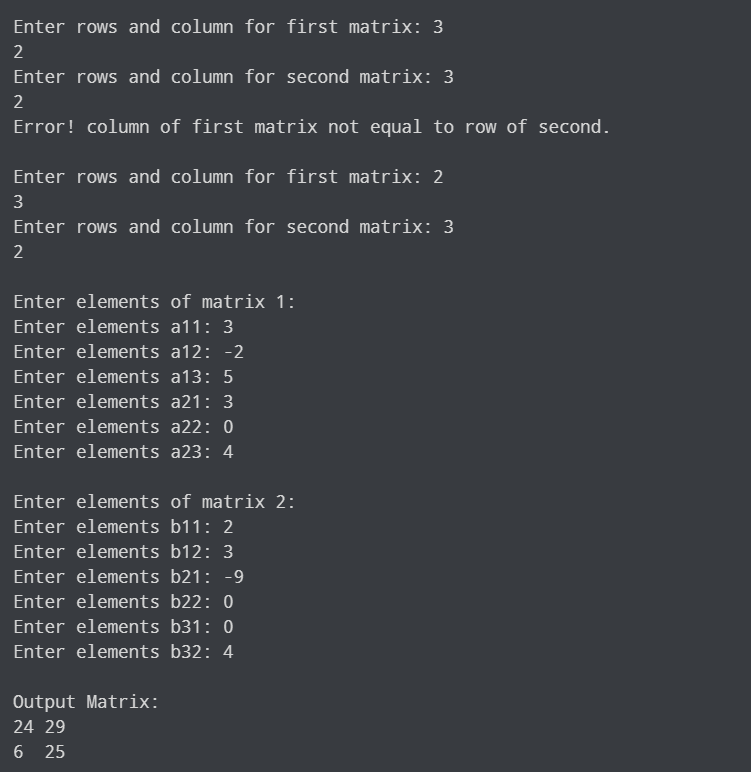
}

multiplyMatrices(row1, col1, m1, row2, col2, m2);

return 0;

}

Output-



Practical No. 5.

**1.WAP to exchange contents of two variables using call by value.**

#include<iostream>

using namespace std;

void exchangeValues(int a, int b) {

int temp = a;

a = b;

b = temp;

cout << "Values inside function after exchange: a = " << a << ", b = " << b << endl;

}

int main() {

int x = 5;

int y = 10;

cout << "Values before exchange: x = " << x << ", y = " << y << endl;

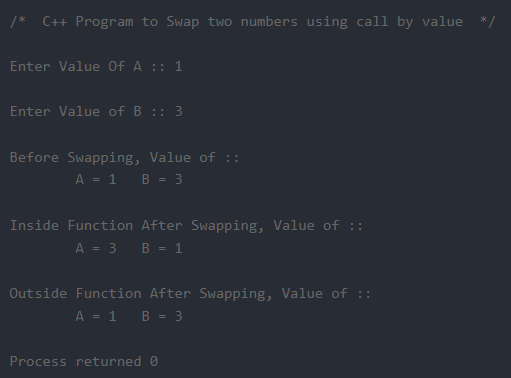
exchangeValues(x, y);

cout << "Values after exchange: x = " << x << ", y = " << y << endl;

return 0;

}

Output-



**2. WAP to exchange contents of two variables using call by reference.**

#include<iostream>

using namespace std;

void exchangeValues(int &a, int &b) {

int temp = a;

a = b;

b = temp;

}

int main() {

int x = 5;

int y = 10;

cout << "Values before exchange: x = " << x << ", y = " << y << endl;

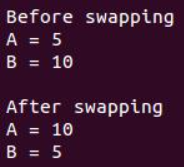
exchangeValues(x, y);

cout << "Values after exchange: x = " << x << ", y = " << y << endl;

return 0;

}

Output-



**3. WAP to create function for addition, subtraction, Multiplication and Division and display the result.**

#include<iostream>

using namespace std;

float addition(float a, float b) {

return a + b;

}

float subtraction(float a, float b) {

return a - b;

}

float multiplication(float a, float b) {

return a \* b;

}

float division(float a, float b) {

if(b == 0) {

throw "Division by zero condition!";

}

return a / b;

}

int main() {

float num1, num2;

char operator\_choice;

cout << "Enter first number: ";

cin >> num1;

cout << "Enter second number: ";

cin >> num2;

cout << "Enter operator (+, -, \*, /): ";

cin >> operator\_choice;

switch(operator\_choice) {

case '+':

cout << "Result: " << addition(num1, num2) << endl;

break;

case '-':

cout << "Result: " << subtraction(num1, num2) << endl;

break;

case '\*':

cout << "Result: " << multiplication(num1, num2) << endl;

break;

case '/':

try {

cout << "Result: " << division(num1, num2) << endl;

} catch (const char\* msg) {

cerr << msg << endl;

}

break;

default:

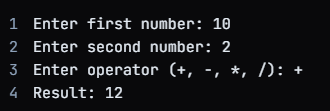
cout << "Invalid operator!" << endl;

}

return 0;

}

Output-



**4. Calculate area of different geometrical figures (circle, rectangle, square, triangle) using function overloading.**

#include<iostream>

#include<cmath>

using namespace std;

float calculateArea(float radius) {

return 3.14159 \* radius \* radius;

}

float calculateArea(float length, float breadth) {

return length \* breadth;

}

float calculateArea(float side) {

return side \* side;

}

float calculateArea(float base, float height) {

return 0.5 \* base \* height;

}

int main() {

float radius, length, breadth, side, base, height;

cout << "Enter radius of circle: ";

cin >> radius;

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

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

cin >> length >> breadth;

cout << "Area of rectangle: " << calculateArea(length, breadth) << endl;

cout << "Enter side of square: ";

cin >> side;

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

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

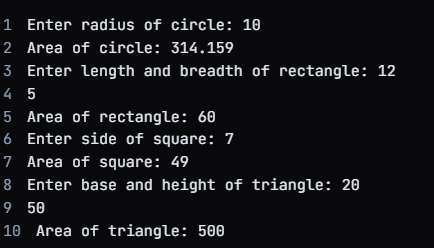
cin >> base >> height;

cout << "Area of triangle: " << calculateArea(base, height) << endl;

return 0;

}

Output-



**5. WAP to maintain the student record which contains Roll number, Name, Marks1, Marks2, Marks3 as data member and getdata(), display() and setdata() as member functions**.

#include<iostream>

#include<cstring>

using namespace std;

class Student {

private:

int Roll;

char Name[20];

float Marks1, Marks2, Marks3;

public:

void getdata() {

cout << "Enter roll number, name, marks1, marks2, marks3: ";

cin >> Roll;

cin.ignore();

cin.getline(Name, 20);

cin >> Marks1 >> Marks2 >> Marks3;

}

void display() {

cout << "\nStudent Details: " << endl;

cout << "Roll number: " << Roll << endl;

cout << "Name: " << Name << endl;

cout << "Marks1: " << Marks1 << endl;

cout << "Marks2: " << Marks2 << endl;

cout << "Marks3: " << Marks3 << endl;

}

void setdata(int r, char\* n, float m1, float m2, float m3) {

Roll = r;

strcpy(Name, n);

Marks1 = m1;

Marks2 = m2;

Marks3 = m3;

}

};

int main() {

Student s1, s2;

s1.getdata();

s1.display();

int Roll;

char Name[20];

float Marks1, Marks2, Marks3;

cout << "Enter roll number, name, marks1, marks2, marks3: ";

cin >> Roll;

cin.ignore();

cin.getline(Name, 20);

cin >> Marks1 >> Marks2 >> Marks3;

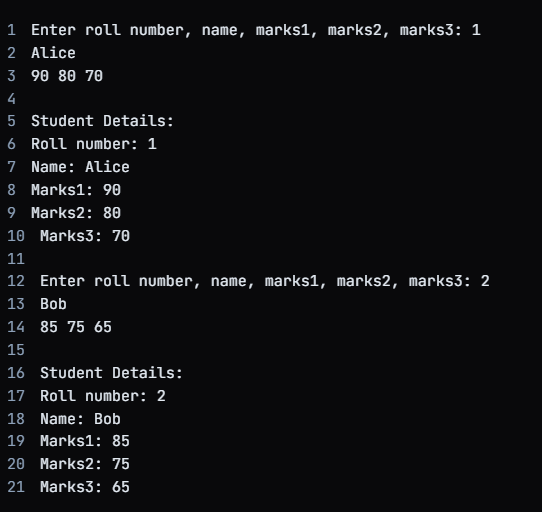
s2.setdata(Roll, Name, Marks1, Marks2, Marks3);

s2.display();

return 0;

}

Output-



Practical 6.

**1.C++ Program to Concatenate and compare Two Strings.**

#include <iostream>

#include <string>

std::string concatenate(const std::string& s1, const std::string& s2) {

return s1 + s2;

}

bool compare(const std::string& s1, const std::string& s2) {

return s1 == s2;

}

int main() {

std::string s1, s2;

std::cout << "Enter first string: ";

std::cin >> s1;

std::cout << "Enter second string: ";

std::cin >> s2;

std::string s3 = concatenate(s1, s2);

std::cout << "Concatenated string: " << s3 << std::endl;

bool result = compare(s1, s2);

if (result) {

std::cout << "Both strings are equal." << std::endl;

} else {

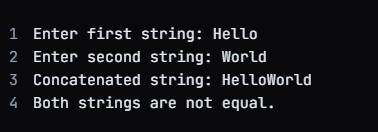
std::cout << "Both strings are not equal." << std::endl;

}

return 0;

}

Output-



**2. C++ Program to Remove Specific Character from the String.**

#include <iostream>

#include <string>

std::string removeChar(std::string str, char c) {

size\_t i = 0, j = 0;

while (i < str.length()) {

if (str[i] != c) {

str[j] = str[i];

j++; // Increment the index of the next position in the result string.

}

i++; // Increment the index of the next character to be inspected.

}

return str.substr(0, j);

}

int main() {

std::string str;

char c;

std::cout << "Enter a string: ";

std::getline(std::cin, str);

std::cout << "Enter a character to be removed: ";

std::cin >> c;

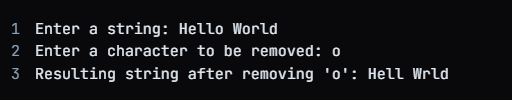
std::string result = removeChar(str, c);

std::cout << "Resulting string after removing '" << c << "': " << result << std::endl;

return 0;

}

Output-



**3. C++ Program to Remove Spaces from a String.**

#include <iostream>

#include <string>

std::string removeSpaces(std::string str) {

str.erase(std::remove(str.begin(), str.end(), ' '), str.end());

return str;

}

int main() {

std::string str;

std::cout << "Enter a string: ";

std::getline(std::cin, str);

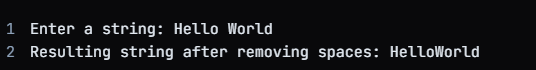
std::string result = removeSpaces(str);

std::cout << "Resulting string after removing spaces: " << result << std::endl;

return 0;

}

Output-



**4. C++ Program to Replace All Spaces in a String with %.**

#include <iostream>

#include <string>

std::string replaceSpaces(std::string str) {

for (size\_t i = 0; i < str.length(); ++i) {

if (str[i] == ' ') {

str[i] = '%';

}

}

return str;

}

int main() {

std::string str;

std::cout << "Enter a string: ";

std::getline(std::cin, str);

std::string result = replaceSpaces(str);

std::cout << "Resulting string after replacing spaces with '%': " << result << std::endl;

return 0;

}

Output-



**5. C++ Program to Reverse Each Word in a String.**

#include <iostream>

#include <sstream>

#include <string>

std::string reverseEachWord(const std::string& str) {

std::istringstream iss(str);

std::string word, reversedWord, output;

while (iss >> word) {

reversedWord = "";

for (int i = word.length() - 1; i >= 0; --i) {

reversedWord += word[i];

}

output += reversedWord + " ";

}

return output.substr(0, output.length() - 1); // remove trailing space

}

int main() {

std::string str;

std::cout << "Enter a string: ";

std::getline(std::cin, str);

std::string reversedString = reverseEachWord(str);

std::cout << "Resulting string after reversing each word: " << reversedString << std::endl;

return 0;

}

Output-



**6. C++ Program to Find the Length of a String.**

#include <iostream>

#include <string>

int main() {

std::string str = "Hello, world!";

std::cout << "The length of the string is: " << str.size() << std::endl;

return 0;

}

Output-



**7. C++ Program to Count Number of Words in a String.**

#include <iostream>

#include <cctype>

int countWords(const std::string& str) {

int count = 0;

bool word = false;

for (char c : str) {

if (std::isalpha(c)) {

if (!word) {

word = true;

count++;

}

} else {

word = false;

}

}

return count;

}

int main() {

std::string str;

std::cout << "Enter a string: ";

std::getline(std::cin, str);

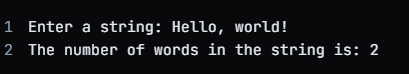
int count = countWords(str);

std::cout << "The number of words in the string is: " << count << std::endl;

return 0;

}

Output-



**8. C++ Program to Swap Two Numbers Without using a Temporary Variable.**

#include <iostream>

void swap(int& a, int& b) {

a = a ^ b;

b = a ^ b;

a = a ^ b;

}

int main() {

int a = 5, b = 10;

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

swap(a, b);

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

return0**;**

**}**

**Output-**

****

Practical No. 7

**1.WAP to increment the employee salaries on the**

**basis of there designation (Manager-5000, General**

**Manager-10000, CEO-20000, worker-2000). Use**

**employee name, id, designation and salary as data**

**member and inc\_sal as member function (Use**

**array of object).**

#include<iostream>

using namespace std;

class Employee {

public:

string name;

int id;

string designation;

int salary;

Employee(string name, int id, string designation, int salary) : name(name), id(id), designation(designation), salary(salary) {}

void inc\_sal() {

if (designation == "Manager") {

salary += 5000;

} else if (designation == "General Manager") {

salary += 10000;

} else if (designation == "CEO") {

salary += 20000;

} else if (designation == "worker") {

salary += 2000;

}

}

void display() {

cout << "Employee Name: " << name << "\n";

cout << "Employee ID: " << id << "\n";

cout << "Employee Designation: " << designation << "\n";

cout << "Employee Salary: " << salary << "\n";

}

};

int main() {

Employee employees[] = {

Employee("John", 1, "Manager", 50000),

Employee("Bob", 2, "General Manager", 80000),

Employee("Alice", 3, "CEO", 150000),

Employee("Tom", 4, "worker", 20000),

};

for (Employee& employee : employees) {

employee.inc\_sal();

employee.display();

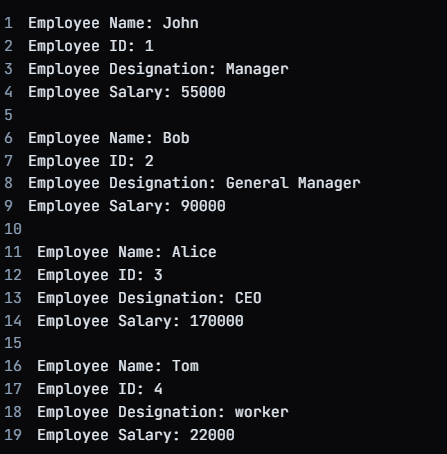
cout << "\n";

}

return 0;

}

Output-



**2. Write a class bank, containing data member: Name**

**of Depositor, A/c type, Type of A/c, Balance**

**amount. Member function: To assign initial value,**

**To deposit an amount, to withdraw an amount after**

**checking the balance (which should be greater than**

**Rs. 500) , To display name & balance.**

#include<iostream>

using namespace std;

class Bank {

private:

string Name\_of\_Depositor;

string A\_c\_type;

string Type\_of\_A\_c;

int Balance\_amount;

public:

Bank() {}

void assign\_initial\_value(string name, string a\_c\_type, string type\_of\_a\_c, int balance) {

Name\_of\_Depositor = name;

A\_c\_type = a\_c\_type;

Type\_of\_A\_c = type\_of\_a\_c;

Balance\_amount = balance;

}

void deposit\_amount(int amount) {

Balance\_amount += amount;

}

void withdraw\_amount(int amount) {

if (Balance\_amount >= 500) {

Balance\_amount -= amount;

} else {

cout << "Insufficient Balance! Cannot withdraw amount.\n";

}

}

void display\_name\_and\_balance() {

cout << "Name of Depositor: " << Name\_of\_Depositor << "\n";

cout << "Account Type: " << A\_c\_type << "\n";

cout << "Account Number: " << Type\_of\_A\_c << "\n";

cout << "Balance Amount: " << Balance\_amount << "\n";

}

};

int main() {

Bank bank;

bank.assign\_initial\_value("Alice", "Savings", "1234567890", 5000);

bank.deposit\_amount(1000);

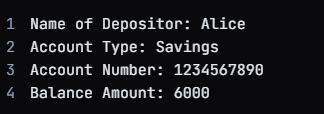
bank.withdraw\_amount(500);

bank.display\_name\_and\_balance();

return 0;

}

Output-



Practical No 8.

**1. WAP to define nested class ‘student\_info’ which 15%**

**contains data members such as name, roll number**

**and sex and also consists of one more class ‘date’**

**,whose data members are day, month and year. The data is to be read from the keyboard & displayed**

**on the screen.**

#include<iostream>

using namespace std;

class Date {

public:

int day, month, year;

void readDate() {

cout << "Enter Date of Birth: ";

cin >> day >> month >> year;

}

void displayDate() {

cout << "Date of Birth: " << day << "/" << month << "/" << year << "\n";

}

};

class StudentInfo {

public:

string name, roll\_number, sex;

Date dob;

void readStudentInfo() {

cout << "Enter Student Details:\n";

cout << "Name: ";

cin.ignore();

getline(cin, name);

cout << "Roll Number: ";

cin >> roll\_number;

cout << "Sex: ";

cin >> sex;

dob.readDate();

}

void displayStudentInfo() {

cout << "Student Details:\n";

cout << "Name: " << name << "\n";

cout << "Roll Number: " << roll\_number << "\n";

cout << "Sex: " << sex << "\n";

dob.displayDate();

}

};

int main() {

StudentInfo student;

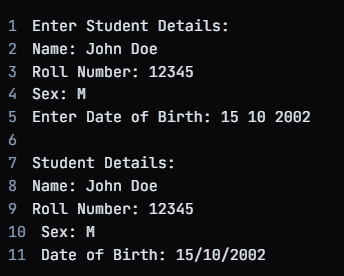
student.readStudentInfo();

student.displayStudentInfo();

return 0;

}

Output-



**2. WAP to generate a series of Fibonacci numbers**

**using copy constructor, where it is defined outside**

**the class using scope resolution operator.**

#include<iostream>

using namespace std;

class Fibonacci {

int n;

int\* fib;

public:

Fibonacci(int n) : n(n) {

fib = new int[n];

}

// Copy Constructor

Fibonacci(const Fibonacci& f) : n(f.n) {

fib = new int[n];

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

fib[i] = f.fib[i];

}

}

~Fibonacci() {

delete[] fib;

}

void generateFibonacciSeries() {

fib[0] = 0;

fib[1] = 1;

for (int i = 2; i < n; i++) {

fib[i] = fib[i - 1] + fib[i - 2];

}

}

void displayFibonacciSeries() {

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

cout << fib[i] << " ";

}

cout << "\n";

}

};

int main() {

int n;

cout << "Enter the number of Fibonacci numbers to generate: ";

cin >> n;

Fibonacci fib1(n);

fib1.generateFibonacciSeries();

fib1.displayFibonacciSeries();

// Invoking Copy Constructor

Fibonacci fib2 = fib1;

fib2.displayFibonacciSeries();

return 0;

}

Output-



Practical 9.

**1)WAP to illustrate single inheritance.**

#include<iostream>

class Animal {

public:

Animal(std::string name, int age) : name(name), age(age) {}

virtual void speak() {

std::cout << "My name is " << name << " and I am " << age << " years old." << std::endl;

}

protected:

std::string name;

int age;

};

class Dog : public Animal {

public:

Dog(std::string name, int age) : Animal(name, age) {}

void speak() override {

std::cout << "My name is " << name << " and I am " << age << " years old. I bark woof!" << std::endl;

}

};

int main() {

Animal \*animal = new Animal("Animal", 10);

Dog \*dog = new Dog("Dog", 5);

animal->speak();

dog->speak();

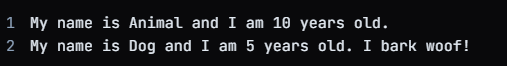
delete animal;

delete dog;

return 0;

}

Output-



**2)WAP to illustrate multiple inheritance**

#include<iostream>

class Engine {

public:

Engine(int cylinders) : cylinders(cylinders) {}

void start() {

std::cout << "Engine with " << cylinders << " cylinders has started." << std::endl;

}

protected:

int cylinders;

};

class BrakeSystem {

public:

BrakeSystem(int size) : size(size) {}

void applyBrake() {

std::cout << "Brake system with " << size << " size has applied brakes." << std::endl;

}

protected:

int size;

};

class Car : public Engine, public BrakeSystem {

public:

Car(int cylinders, int brakeSize) : Engine(cylinders), BrakeSystem(brakeSize) {}

void start() {

Engine::start();

BrakeSystem::applyBrake();

}

};

int main() {

Car \*car = new Car(6, 32);

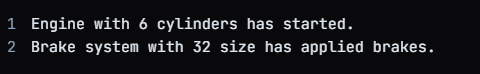
car->start();

delete car;

return 0;

}

Output-



**3)WAP to illustrate multilevel inheritance**

#include<iostream>

class Base {

public:

Base(int value) : value(value) {}

protected:

int value;

};

class Derived1 : public Base {

public:

Derived1(int value) : Base(value) {}

void display() {

std::cout << "Value from Derived1: " << value << std::endl;

}

};

class Derived2 : public Derived1 {

public:

Derived2(int value) : Derived1(value) {}

void display() {

std::cout << "Value from Derived2: " << value << std::endl;

}

};

int main() {

Derived2 \*obj = new Derived2(42);

obj->display();

delete obj;

return 0;

}

Output-



**4)WAP to illustrate hierarchical inheritance**

#include<iostream>

class Base {

public:

Base(int value) : value(value) {}

protected:

int value;

};

class Derived1 : public Base {

public:

Derived1(int value) : Base(value) {}

void display() {

std::cout << "Value from Derived1: " << value << std::endl;

}

};

class Derived2 : public Derived1 {

public:

Derived2(int value) : Derived1(value) {}

void display() {

std::cout << "Value from Derived2: " << value << std::endl;

}

};

class Derived3 : public Derived2 {

public:

Derived3(int value) : Derived2(value) {}

void display() {

std::cout << "Value from Derived3: " << value << std::endl;

}

};

int main() {

Derived3 \*obj = new Derived3(42);

obj->display();

delete obj;

return 0;

}

Output-



**5)WAP to illustrate hybrid inheritance**

#include<iostream>

class Base {

public:

Base(int value) : value(value) {}

protected:

int value;

};

class Derived1 : public Base {

public:

Derived1(int value) : Base(value) {}

void display() {

std::cout << "Value from Derived1: " << value << std::endl;

}

};

class Derived2 : public Derived1 {

public:

Derived2(int value) : Derived1(value) {}

void display() {

std::cout << "Value from Derived2: " << value << std::endl;

}

};

class Derived3 : public Base, public Derived2 {

public:

Derived3(int value) : Base(value), Derived2(value) {}

void display() {

std::cout << "Value from Derived3: " << value << std::endl;

}

};

int main() {

Derived3 \*obj = new Derived3(42);

obj->display();

delete obj;

return 0;

}

Output-



**6)Create a class Employee. Derive 3 classes from this class namely, Programmer, Analyst & Project Leader. Take attributes and operations on your own. WAP to implement this with array of pointers.**

#include<iostream>

using namespace std;

class Employee {

public:

Employee(int id) : id(id) {}

protected:

int id;

};

class Programmer : public Employee {

public:

Programmer(int id) : Employee(id) {}

void display() {

cout << "Employee ID: " << id << endl;

cout << "Role: Programmer" << endl;

}

};

class Analyst : public Employee {

public:

Analyst(int id) : Employee(id) {}

void display() {

cout << "Employee ID: " << id << endl;

cout << "Role: Analyst" << endl;

}

};

class ProjectLeader : public Employee {

public:

ProjectLeader(int id) : Employee(id) {}

void display() {

cout << "Employee ID: " << id << endl;

cout << "Role: Project Leader" << endl;

}

};

int main() {

int n;

cout << "Enter number of employees: ";

cin >> n;

Employee\*\* employees = new Employee\*[n];

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

int role, id;

cout << "Enter role (1 - Programmer, 2 - Analyst, 3 - Project Leader): ";

cin >> role;

cout << "Enter employee ID: ";

cin >> id;

switch (role) {

case 1:

employees[i] = new Programmer(id);

break;

case 2:

employees[i] = new Analyst(id);

break;

case 3:

employees[i] = new ProjectLeader(id);

break;

default:

cout << "Invalid role." << endl;

break;

}

}

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

employees[i]->display();

delete employees[i];

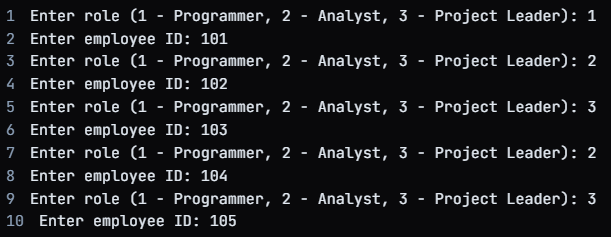
}

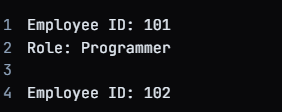
delete[] employees;

return 0;

}

Output-





**7)Create two classes namely Employee and Qualification. Using multiple inheritance derive two classes Scientist and Manager. Take suitable attributes & operations**

#include<iostream>

using namespace std;

class Employee {

public:

Employee(int id) : id(id) {}

protected:

int id;

};

class Qualification {

public:

Qualification(string q) : qualification(q) {}

protected:

string qualification;

};

class Scientist : public Employee, public Qualification {

public:

Scientist(int id, string q) : Employee(id), Qualification(q) {}

void display() {

cout << "Employee ID: " << id << endl;

cout << "Role: Scientist" << endl;

cout << "Qualification: " << qualification << endl;

}

};

class Manager : public Employee, public Qualification {

public:

Manager(int id, string q) : Employee(id), Qualification(q) {}

void display() {

cout << "Employee ID: " << id << endl;

cout << "Role: Manager" << endl;

cout << "Qualification: " << qualification << endl;

}

};

int main() {

int n;

cout << "Enter number of employees: ";

cin >> n;

Employee\*\* employees = new Employee\*[n];

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

int role, id;

string qualification;

cout << "Enter role (1 - Scientist, 2 - Manager): ";

cin >> role;

cout << "Enter employee ID: ";

cin >> id;

cout << "Enter qualification: ";

cin >> qualification;

switch (role) {

case 1:

employees[i] = new Scientist(id, qualification);

break;

case 2:

e qmployees[i] = new Manager(id, qualification);

break;

default:

cout << "Invalid role." << endl;

break;

}

}

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

employees[i]->display();

delete employees[i];

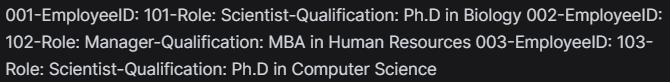
}

delete[] employees;

return 0;

}

Output-



Practical 10.

**1)WAP to implement ‘this’ keyword**

#include <iostream>

using namespace std;

class GFG

{

string name;

public:

Abdullah(string name)

{ this->name = name;

}

void display()

{

cout << name << endl;

}

};

// Driver code

int main()

{

Abdullah abdullah("Practical");

abdullah.display();

return 0;

}

Output-



**2)WAP to implement static keyword**

#include<iostream>

class MyClass {

public:

static int staticVariable;

static void staticMethod() {

std::cout << "Static method called!" << std::endl;

}

};

int MyClass::staticVariable = 0;

int main() {

std::cout << "Static variable value: " << MyClass::staticVariable << std::endl;

MyClass::staticMethod();

MyClass obj;

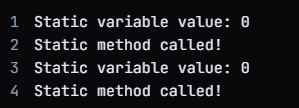
std::cout << "Static variable value: " << obj.staticVariable << std::endl;

obj.staticMethod();

return 0;

}

Output-

****

**3)WAP to create a class student containing Name & class as parameters, create another class marks which inherit student class taking own data members as mark1 & mark2 &show result.**

#include<iostream>

using namespace std;

class Student {

string Name;

int Class;

public:

Student(string n, int c) : Name(n), Class(c) {}

string getName() {

return Name;

}

int getClass() {

return Class;

}

};

class Marks : public Student {

int mark1, mark2;

public:

Marks(string n, int c, int m1, int m2) : Student(n, c), mark1(m1), mark2(m2) {}

void showResult() {

cout << "Name: " << getName() << endl;

cout << "Class: " << getClass() << endl;

cout << "Mark 1: " << mark1 << endl;

cout << "Mark 2: " << mark2 << endl;

}

};

int main() {

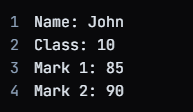
Marks obj("John", 10, 85, 90);

obj.showResult();

return 0;

}

Output-



Practical 11.

**1)Write a class string to compare two strings, overload (= =) operator.**

#include<iostream>

using namespace std;

class MyString {

string str;

public:

MyString(string s) : str(s) {}

bool operator==(const MyString& other) const {

return str == other.str;

}

string getString() const {

return str;

}

};

int main() {

MyString str1("Hello");

MyString str2("Hello");

MyString str3("World");

if (str1 == str2) {

cout << str1.getString() << " and " << str2.getString() << " are equal." << endl;

} else {

cout << str1.getString() << " and " << str2.getString() << " are not equal." << endl;

}

if (str1 == str3) {

cout << str1.getString() << " and " << str3.getString() << " are equal." << endl;

} else {

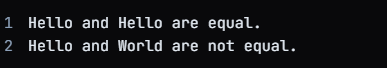
cout << str1.getString() << " and " << str3.getString() << " are not equal." << endl;

}

return 0;

}

Output-



**2)Write a class to concatenate two strings, overload (+) operator.**

#include<iostream>

using namespace std;

class MyString {

string str;

public:

MyString(string s) : str(s) {}

MyString operator+(const MyString& other) const {

return MyString(str + other.str);

}

string getString() const {

return str;

}

};

int main() {

MyString str1("Hello");

MyString str2("World");

MyString str3("HelloWorld");

MyString str4 = str1 + str2;

if (str4 == str3) {

cout << str1.getString() << " and " << str2.getString() << " concatenated are equal to " << str3.getString() << "." << endl;

} else {

cout << str1.getString() << " and " << str2.getString() << " concatenated are not equal to " << str3.getString() << "." << endl;

}

return 0;

}

Output-



**3)Create a class item, having two data members x & y, overload ‘-‘(unary operator) to change the sign of x and y.**

#include<iostream>

using namespace std;

class Item {

int x, y;

public:

Item(int a, int b) : x(a), y(b) {}

void display() const {

cout << "(" << x << ", " << y << ")" << endl;

}

Item operator-() const {

return Item(-x, -y);

}

};

int main() {

Item item1(3, 4);

Item item2(-3, -4);

cout << "Before applying unary '-': ";

item1.display();

Item item3 = -item1;

cout << "After applying unary '-': ";

item3.display();

return 0;

}

Output-

