PROGRAM 1]

Write a C++ program to perform addition of two number using class and object.

PROPOSED ALGORITHM :

* Step 1 : Define the class name ‘Adder’.
* Step 2 : Using private member to store number ‘num1’ and ‘num2’.
* Step 3: Public member function to set the values of ‘num1’ and ‘num2’.
* Step 4 : Using public member function ‘add’ to perform addition and return the result.
* Step 5 : In main function , create an object of the Adder class.
* Step 6 : Set the values of num1 and num2 using the ‘setNumbers’ function.
* Step 7 : Perform addition and display the result.

#include<iostream>

class Adder {

private:

int num1;

int num2;

public:

void setNumbers(int a, int b) {

num1 = a;

num2 = b;

}

int add() {

return num1 + num2;

}

};

int main() {

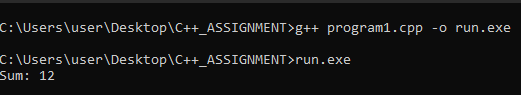
Adder myAdder;

myAdder.setNumbers(5, 7);

std::cout << "Sum: " << myAdder.add() << std::endl;

return 0;

}

OUTPUT :

PROGRAM 2]

Write a program to input 20 arbitrary numbers in one-dimensional array. Calculate Frequency of each number. Print the number and its frequency in a tabular form.

PROPOSED ALGORITHM :

* Step 1 : Create an array of size 20 to store the input numbers.
* Step 2 : Use a loop to input 20 arbitrary numbers into array.
* Step 3: Create an array to store frequency and initialize to 0.
* Step 4 : Calculate the frequency of each number.
* Step 5 : Print the header for the table using cout.
* Step 6 :print number and their frequencies in tabular form using ‘for’ loop.
* Step 7 : Avoid printing duplicates using ‘If-else’ condition structure.

#include<iostream>

#include<iomanip>

int main() {

int numbers[20];

std::cout << "Enter 20 arbitrary numbers:" << std::endl;

for (int i = 0; i < 20; ++i) {

std::cin >> numbers[i];

}

int frequency[20] = {0};

for (int i = 0; i < 20; ++i) {

for (int j = 0; j < 20; ++j) {

if (numbers[j] == numbers[i]) {

frequency[i]++;

}

}

}

std::cout << "\nNumber\tFrequency" << std::endl;

for (int i = 0; i < 20; ++i) {

if (frequency[i] != 0) {

std::cout << std::setw(3) << numbers[i] << "\t" << std::setw(3) << frequency[i] << std::endl;

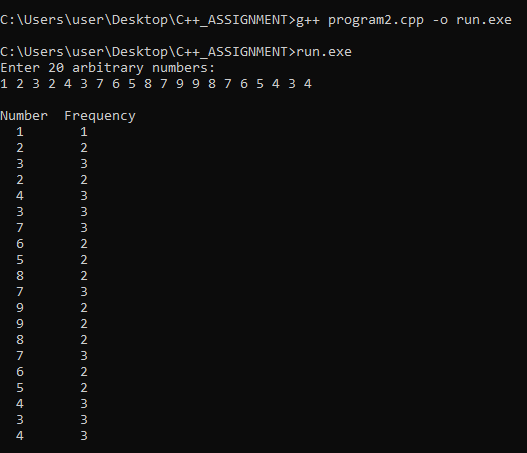
}

}

return 0;

}

OUTPUT :



PROGRAM 3]

Write a C++ program to perform addition of two number using function.

PROPOSED ALGORITHM :

* Step 1 :Start with including preprocessor header files.
* Step 2 : Declare the function ‘addNumbers’.
* Step 3: Declare a variable sum to store the result of addition.
* Step 4 : Add num1 and num2 and store the result in the sum variable.
* Step 5 : Return the value of sum.
* Step 6 : In main function, Declare variables num1, num2, result to store data.
* Step 7 : Prompt user to enter two number for addition.
* Step 8 : call the ‘addNumber’ function with num1 and num2 arguments and store the answer in result variable and display.

#include<iostream>

int addNumbers(int num1, int num2) {

int sum;

sum = num1 + num2;

return sum;

}

int main() {

int num1, num2, result;

std::cout << "Enter the first number: ";

std::cin >> num1;

std::cout << "Enter the second number: ";

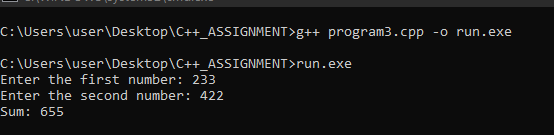
std::cin >> num2;

result = addNumbers(num1, num2);

std::cout << "Sum: " << result << std::endl;

return 0;

}

OUTPUT :

PROGRAM 4]

Write a C++ program for dynamic initialization of variable.

PROPOSED ALGORITHM :

* Step 1 : Start with including preprocessor header files.
* Step 2 : Declare the pointer variable of desired data type.
* Step 3 : Prompt user to enter the value for the variable.
* Step 4 : Dynamically allocate memory for the variable.
* Step 5 : Initialize the dynamically allocated variable.
* Step 6 : Display the value of the dynamically initialized variable.
* Step 7 : Release the dynamically allocated memory.
* Step 8 : End.

#include<iostream>

int main() {

int\* dynamicVariable;

std::cout << "Enter the value for the variable: ";

dynamicVariable = new int;

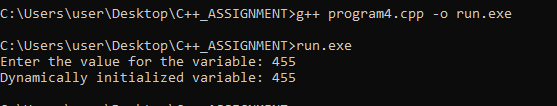
std::cin >> \*dynamicVariable;

std::cout << "Dynamically initialized variable: " << \*dynamicVariable << std::endl;

delete dynamicVariable;

return 0;

}

OUTPUT :

PROGRAM 5]

Write a C++ Program that uses a class where the member functions are defined inside the class.

PROPOSED ALGORITHM :

* Step 1 : Start with including preprocessor header files.
* Step 2 : Declare the calculator class.
* Step 3 : Declare private member variable.
* Step 4 : Declare public member function.
* Step 5 : Define the member function inside the class.
* Step 6 : Create an object of the calculator class.
* Step 7 : Display a menu.
* Step 8 : call the corresponding function based on input.
* Step 9: End.

#include<iostream>

class Calculator {

private:

double num1, num2;

public:

void addition() {

std::cout << "Sum: " << num1 + num2 << std::endl;

}

void subtraction() {

std::cout << "Difference: " << num1 - num2 << std::endl;

}

void multiplication() {

std::cout << "Product: " << num1 \* num2 << std::endl;

}

void division() {

if (num2 != 0) {

std::cout << "Quotient: " << num1 / num2 << std::endl;

} else {

std::cout << "Error: Division by zero." << std::endl;

}

}

void setNumbers(double a, double b) {

num1 = a;

num2 = b;

}};

int main() {

Calculator myCalculator;

int choice;

do {

std::cout << "\nCalculator Menu:\n";

std::cout << "1. Addition\n";

std::cout << "2. Subtraction\n";

std::cout << "3. Multiplication\n";

std::cout << "4. Division\n";

std::cout << "5. Exit\n";

std::cout << "Enter your choice: ";

std::cin >> choice;

switch (choice) {

case 1:

case 2:

case 3:

case 4:

double a, b;

std::cout << "Enter two numbers: ";

std::cin >> a >> b;

myCalculator.setNumbers(a, b);

switch (choice) {

case 1:

myCalculator.addition();

break;

case 2:

myCalculator.subtraction();

break;

case 3:

myCalculator.multiplication();

break;

case 4:

myCalculator.division();

break;

}

break;

case 5:

std::cout << "Exiting the program.\n";

break;

default:

std::cout << "Invalid choice. Please enter a valid option.\n";

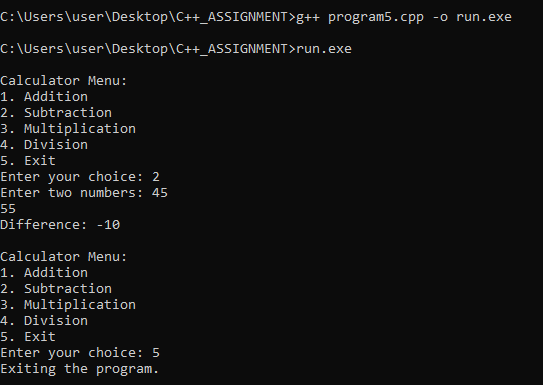
}

} while (choice != 5);

return 0;

}

OUTPUT :



PROGRAM 6]

Write a program that uses a class where the member functions are defined outside a class.

PROPOSED ALGORITHM :

* Step 1 : Start with including preprocessor header files.
* Step 2 : Declare the calculator class.
* Step 3 : Declare private member variable.
* Step 4 : Declare public member function.
* Step 5 : Define the member function outside the class using scope resolution.
* Step 6 : Create an object of the calculator class.
* Step 7 : Display a menu.
* Step 8 : call the corresponding function based on input.
* Step 9: End.

#include<iostream>

class Calculator {

private:

double num1, num2;

public:

void setNumbers(double a, double b);

void addition();

void subtraction();

void multiplication();

void division();

};

void Calculator::setNumbers(double a, double b) {

num1 = a;

num2 = b;

}

void Calculator::addition() {

std::cout << "Sum: " << num1 + num2 << std::endl;

}

void Calculator::subtraction() {

std::cout << "Difference: " << num1 - num2 << std::endl;

}

void Calculator::multiplication() {

std::cout << "Product: " << num1 \* num2 << std::endl;

}

void Calculator::division() {

if (num2 != 0) {

std::cout << "Quotient: " << num1 / num2 << std::endl;

} else {

std::cout << "Error: Division by zero." << std::endl;

}

}

int main() {

Calculator myCalculator;

int choice;

do {

std::cout << "\nCalculator Menu:\n";

std::cout << "1. Addition\n";

std::cout << "2. Subtraction\n";

std::cout << "3. Multiplication\n";

std::cout << "4. Division\n";

std::cout << "5. Exit\n";

std::cout << "Enter your choice: ";

std::cin >> choice;

switch (choice) {

case 1:

case 2:

case 3:

case 4:

double a, b;

std::cout << "Enter two numbers: ";

std::cin >> a >> b;

myCalculator.setNumbers(a, b);

switch (choice) {

case 1:

myCalculator.addition();

break;

case 2:

myCalculator.subtraction();

break;

case 3:

myCalculator.multiplication();

break;

case 4:

myCalculator.division();

break;

}

break;

case 5:

std::cout << "Exiting the program.\n";

break;

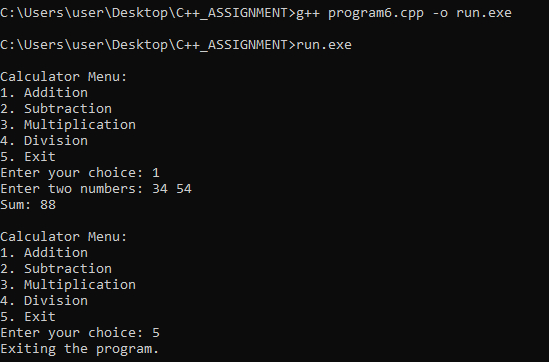
default:

std::cout << "Invalid choice. Please enter a valid option.\n";

}

} while (choice != 5); return 0;}

OUTPUT :



PROGRAM 7]

Write a C++ program to calculate simple interest using function.

PROPOSED ALGORITHM :

* Step 1 : Start with including preprocessor header files.
* Step 2 : Declare the function ‘calculateSimpleInterest’.
* Step 3 : Declare a variable ‘simpleInterest’ to store the result.
* Step 4 : Calculate simple interest using formula.
* Step 5 : Return the value of ‘simpleInterest’.
* Step 6 : In main function, Declare variable principal, rate, time, and result.
* Step 7 : Initialize variable and call function ‘calculateSimpleInterest’ with arguments .
* Step 8 : Display result.

#include<iostream>

float calculateSimpleInterest(float principal, float rate, float time) {

float simpleInterest;

simpleInterest = (principal \* rate \* time) / 100;

return simpleInterest;

}

int main() {

float principal, rate, time, result;

std::cout << "Enter the principal amount: ";

std::cin >> principal;

std::cout << "Enter the rate of interest: ";

std::cin >> rate;

std::cout << "Enter the time (in years): ";

std::cin >> time;

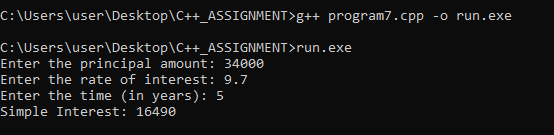
result = calculateSimpleInterest(principal, rate, time);

std::cout << "Simple Interest: " << result << std::endl;

return 0;

}

OUTPUT :



PROGRAM 8]

Write a C++ program to calculate simple interest using class and object.

code with comments for better understanding :

#include<iostream>

// Step 1: Declare a class named SimpleInterestCalculator

class SimpleInterestCalculator {

private:

// Step 2: Declare private member variables for principal, rate, and time

float principal;

float rate;

float time;

public:

// Step 3: Declare public member functions for input, calculation, and output

void inputValues() {

// a. Prompt the user to enter the principal amount, rate of interest, and time

std::cout << "Enter the principal amount: ";

std::cin >> principal;

std::cout << "Enter the rate of interest: ";

std::cin >> rate;

std::cout << "Enter the time (in years): ";

std::cin >> time;

}

float calculateSimpleInterest() {

// b. Calculate simple interest using the formula

return (principal \* rate \* time) / 100;

}

void displayResult(float simpleInterest) {

// c. Display the calculated simple interest

std::cout << "Simple Interest: " << simpleInterest << std::endl;

}

};

// Step 4: In the main function

int main() {

// a. Create an object of the SimpleInterestCalculator class

SimpleInterestCalculator calculator;

// b. Call the inputValues function to get user input

calculator.inputValues();

// c. Call the calculateSimpleInterest function to perform the calculation

float result = calculator.calculateSimpleInterest();

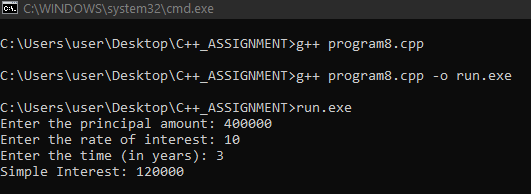
// d. Call the displayResult function to display the result

calculator.displayResult(result);

// Step 5: End

return 0;

}

OUTPUT :

PROGRAM 9]

Write a C++ program to print weekday using if else statement.

code with comments for better understanding :

#include<iostream>

int main() {

// Step 1: Declare a variable to store the day of the week (1 to 7)

int day;

// Step 2: Prompt the user to enter a number representing the day of the week

std::cout << "Enter a number (1 to 7) representing the day of the week: ";

std::cin >> day;

// Step 3: Use if-else statements to determine the day and print the corresponding name

if (day == 1) {

std::cout << "Sunday" << std::endl;

} else if (day == 2) {

std::cout << "Monday" << std::endl;

} else if (day == 3) {

std::cout << "Tuesday" << std::endl;

} else if (day == 4) {

std::cout << "Wednesday" << std::endl;

} else if (day == 5) {

std::cout << "Thursday" << std::endl;

} else if (day == 6) {

std::cout << "Friday" << std::endl;

} else if (day == 7) {

std::cout << "Saturday" << std::endl;

} else {

// Step 4: Handle the case when the entered number is not in the valid range

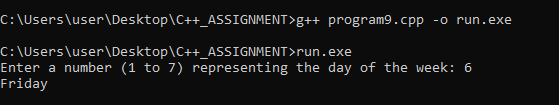
std::cout << "Invalid input. Please enter a number between 1 and 7." << std::endl;

}

// Step 5: End

return 0;

}

OUTPUT :

PROGRAM 10]

Write a C++ program to print weekday using switch-case statement.

code with comments for better understanding :

#include<iostream>

int main() {

// Step 1: Declare a variable to store the day of the week (1 to 7)

int day;

// Step 2: Prompt the user to enter a number representing the day of the week

std::cout << "Enter a number (1 to 7) representing the day of the week: ";

std::cin >> day;

// Step 3: Use a switch statement to determine the day and print the corresponding name

switch(day) {

case 1:

std::cout << "Sunday" << std::endl;

break;

case 2:

std::cout << "Monday" << std::endl;

break;

case 3:

std::cout << "Tuesday" << std::endl;

break;

case 4:

std::cout << "Wednesday" << std::endl;

break;

case 5:

std::cout << "Thursday" << std::endl;

break;

case 6:

std::cout << "Friday" << std::endl;

break;

case 7:

std::cout << "Saturday" << std::endl;

break;

default:

/\* Step 4: Handle the case when the entered number is not in the valid range\*/

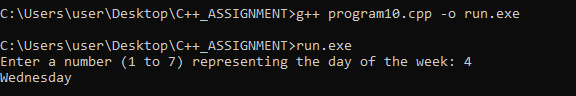
std::cout << "Invalid input. Please enter a number between 1 and 7." << std::endl;

}

// Step 5: End

return 0;

}

OUTPUT :

PROGRAM 11]

Write a C++ program to design simple calculator to perform arithmetic operation.

code with comments for better understanding :

#include <iostream>

using namespace std;

int main() {

// Declare variables to store user input and result

double num1, num2, result;

char op; // Operator (+, -, \*, /)

// Get user input

cout << "Enter first number: ";

cin >> num1;

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

cin >> op;

cout << "Enter second number: ";

cin >> num2;

// Perform arithmetic operation based on the operator

switch (op) {

case '+':

result = num1 + num2;

break;

case '-':

result = num1 - num2;

break;

case '\*':

result = num1 \* num2;

break;

case '/':

// Check if the divisor is not zero

if (num2 != 0) {

result = num1 / num2;

} else {

cout << "Error: Division by zero is undefined." << endl;

return 1; // Exit the program with an error code

}

break;

default:

cout << "Error: Invalid operator." << endl;

return 1; // Exit the program with an error code

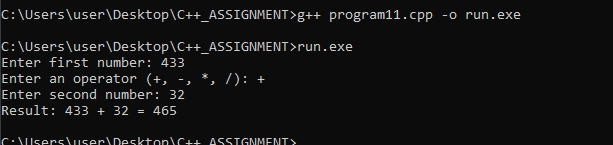
}

// Display the result

cout << "Result: " << num1 << " " << op << " " << num2 << " = " << result << endl;

return 0; // Exit the program successfully

}

Output

PROGRAM 12]

Write a c++ program to design mark sheet of 1st SEM.

code with comments for better understanding :

#include <iostream>

#include <iomanip> // For setting precision in output

using namespace std;

int main() {

// Declare variables to store subject marks

double mathMarks, physicsMarks, chemistryMarks;

// Get input for each subject

cout << "Enter Mathematics marks (out of 100): ";

cin >> mathMarks;

cout << "Enter Physics marks (out of 100): ";

cin >> physicsMarks;

cout << "Enter Chemistry marks (out of 100): ";

cin >> chemistryMarks;

// Calculate total and average marks

double totalMarks = mathMarks + physicsMarks + chemistryMarks;

double averageMarks = totalMarks / 3.0;

// Display the mark sheet

cout << "\nMark Sheet\n";

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

cout << setw(15) << "Subject" << setw(10) << "Marks\n";

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

cout << setw(15) << "Mathematics" << setw(10) << mathMarks << endl;

cout << setw(15) << "Physics" << setw(10) << physicsMarks << endl;

cout << setw(15) << "Chemistry" << setw(10) << chemistryMarks << endl;

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

cout << setw(15) << "Total" << setw(10) << totalMarks << endl;

cout << setw(15) << "Average" << setw(10) << fixed << setprecision(2) << averageMarks << endl;

// Display grades based on average marks

if (averageMarks >= 90) {

cout << "Grade: A\n";

} else if (averageMarks >= 80) {

cout << "Grade: B\n";

} else if (averageMarks >= 70) {

cout << "Grade: C\n";

} else if (averageMarks >= 60) {

cout << "Grade: D\n";

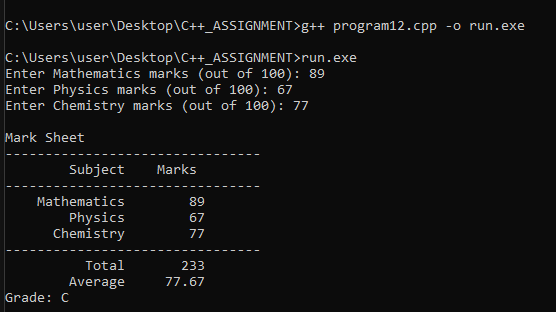
} else {

cout << "Grade: F\n";

}

return 0; // Exit the program successfully

}

Output :

PROGRAM 13]

Write a c++ program to print weekday using enum.

code with comments for better understanding :

#include <iostream>

using namespace std;

// Enumeration for weekdays

enum Weekday {

MONDAY,

TUESDAY,

WEDNESDAY,

THURSDAY,

FRIDAY,

SATURDAY,

SUNDAY

};

int main() {

// Declare a variable of type Weekday

Weekday day;

// Get user input for the day (assuming 0 for Monday, 1 for Tuesday, and so on)

cout << "Enter a number (0-6) to represent a day of the week: ";

int userInput;

cin >> userInput;

// Validate user input

if (userInput < 0 || userInput > 6) {

cout << "Invalid input. Please enter a number between 0 and 6." << endl;

return 1; // Exit the program with an error code

}

// Assign the user input to the variable 'day'

day = static\_cast<Weekday>(userInput);

// Print the corresponding weekday

switch (day) {

case MONDAY:

cout << "The day is Monday." << endl;

break;

case TUESDAY:

cout << "The day is Tuesday." << endl;

break;

case WEDNESDAY:

cout << "The day is Wednesday." << endl;

break;

case THURSDAY:

cout << "The day is Thursday." << endl;

break;

case FRIDAY:

cout << "The day is Friday." << endl;

break;

case SATURDAY:

cout << "The day is Saturday." << endl;

break;

case SUNDAY:

cout << "The day is Sunday." << endl;

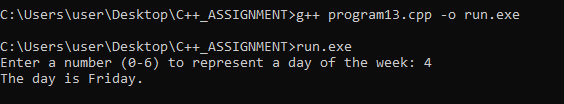
break;

}

return 0; // Exit the program successfully

}

Output :



PROGRAM 14]

Write a c++ program to find largest number using conditional operator

code with comments for better understanding :

#include <iostream>

using namespace std;

int main() {

// Declare variables to store three numbers

int num1, num2, num3;

// Get user input for three numbers

cout << "Enter the first number: ";

cin >> num1;

cout << "Enter the second number: ";

cin >> num2;

cout << "Enter the third number: ";

cin >> num3;

// Use the conditional operator to find the largest number

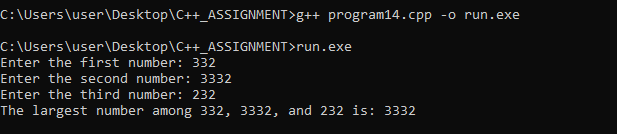
int largestNumber = (num1 > num2) ? ((num1 > num3) ? num1 : num3) : ((num2 > num3) ? num2 : num3);

// Display the result

cout << "The largest number among " << num1 << ", " << num2 << ", and " << num3 << " is: " << largestNumber << endl;

return 0; // Exit the program successfully

}

Output :

PROGRAM 15]

Write a c++ program to find the greatest number of three using goto statement.

code with comments for better understanding :

#include <iostream>

using namespace std;

int main() {

// Declare variables to store three numbers

int num1, num2, num3;

// Get user input for three numbers

cout << "Enter the first number: ";

cin >> num1;

cout << "Enter the second number: ";

cin >> num2;

cout << "Enter the third number: ";

cin >> num3;

// Using goto to find the greatest number

int greatestNumber = num1;

if (num2 > greatestNumber)

greatestNumber = num2;

if (num3 > greatestNumber)

greatestNumber = num3;

// Display the result

cout << "The greatest number among " << num1 << ", " << num2 << ", and " << num3 << " is: " << greatestNumber << endl;

return 0; // Exit the program successfully

}

Output : 15

PROGRAM 16]

Write a c++ program to find the greatest number of three using switch statement.

code with comments for better understanding :

#include <iostream>

using namespace std;

int main() {

// Declare variables to store three numbers

int num1, num2, num3;

// Get user input for three numbers

cout << "Enter the first number: ";

cin >> num1;

cout << "Enter the second number: ";

cin >> num2;

cout << "Enter the third number: ";

cin >> num3;

int greatestNumber;

// Using a simulated switch statement to find the greatest number

switch (0) {

case (num1 - num2):

case (num1 - num3):

greatestNumber = num1;

break;

case (num2 - num1):

case (num2 - num3):

greatestNumber = num2;

break;

case (num3 - num1):

case (num3 - num2):

greatestNumber = num3;

break;

default:

cout << "Error in switch statement." << endl;

return 1; // Exit the program with an error code

}

// Display the result

cout << "The greatest number among " << num1 << ", " << num2 << ", and " << num3 << " is: " << greatestNumber << endl;

return 0; // Exit the program successfully

}

Output 16

PROGRAM 17]

Write a c++ program to print table of two using goto statement.

code with comments for better understanding :

#include<iostream>

using namespace std;

int main() {

int i = 1;

// Using goto statement along with for loop

for (;;) {

cout << "2 x " << i << " = " << 2 \* i << endl;

i++;

if (i > 10)

goto end;

}

end:

return 0;

}

Output 17 :

PROGRAM 18]

Write a c++ program to find winner of candidate for election.

code with comments for better understanding :

#include <iostream>

#include <unordered\_map>

#include <algorithm>

using namespace std;

int main() {

// Declare variables

int totalVotes;

unordered\_map<string, int> candidateVotes; // Map to store candidate names and their votes

// Get the total number of votes

cout << "Enter the total number of votes: ";

cin >> totalVotes;

// Input votes for each candidate

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

string candidateName;

cout << "Enter the name of the candidate for vote " << i + 1 << ": ";

cin >> candidateName;

// Increment the vote count for the candidate

candidateVotes[candidateName]++;

}

// Find the winner by determining the candidate with the maximum votes

auto winner = max\_element(candidateVotes.begin(), candidateVotes.end(),

[](const auto& a, const auto& b) {

return a.second < b.second;

}

);

// Display the winner

cout << "\nThe winner of the election is: " << winner->first << " with " << winner->second << " votes." << endl;

return 0; // Exit the program successfully

}

Output 18 :

PROGRAM 19]

Write a c++ program to check whether given no. is prime or not.

code with comments for better understanding :

#include <iostream>

#include <cmath>

using namespace std;

int main() {

// Declare variable to store the user input

int num;

// Get user input

cout << "Enter a number: ";

cin >> num;

// Check if the number is less than 2

if (num < 2) {

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

return 0; // Exit the program successfully

}

// Check for factors from 2 to the square root of the number

for (int i = 2; i <= sqrt(num); ++i) {

if (num % i == 0) {

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

return 0; // Exit the program successfully

}

}

// If no factors were found, the number is prime

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

return 0; // Exit the program successfully

}

Output 19:

PROGRAM 20]

Write a program to evaluate the sum of 1+2+3+…………..+n while loop.

code with comments for better understanding

#include <iostream>

using namespace std;

int main() {

// Declare variables

int n, sum = 0, i = 1;

// Get user input for the value of n

cout << "Enter the value of n: ";

cin >> n;

// Use a while loop to calculate the sum

while (i <= n) {

sum += i; // Add the current value of i to the sum

i++; // Increment i for the next iteration

}

// Display the result

cout << "The sum of the series 1 + 2 + 3 + ... + " << n << " is: " << sum << endl;

return 0; // Exit the program successfully

}

Output 20 :

PROGRAM 22]

Write a c++ program to generate 3x3 matrix.

code with comments for better understanding

#include <iostream>

using namespace std;

int main() {

// Declare a 3x3 matrix

int matrix[3][3];

// Get user input for the elements of the matrix

cout << "Enter the elements of the 3x3 matrix:" << endl;

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

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

cout << "Enter element at position (" << i + 1 << ", " << j + 1 << "): ";

cin >> matrix[i][j];

}

}

// Display the generated matrix

cout << "\nThe generated 3x3 matrix is:" << endl;

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

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

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

}

cout << endl;

}

return 0; // Exit the program successfully

}

Output 22 :

PROGRAM 23]

Write a c++ program for addition of 3x3 matrix.

code with comments for better understanding

#include <iostream>

using namespace std;

int main() {

// Declare two 3x3 matrices

int matrix1[3][3], matrix2[3][3], resultMatrix[3][3];

// Get user input for the elements of the first matrix

cout << "Enter the elements of the first 3x3 matrix:" << endl;

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

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

cout << "Enter element at position (" << i + 1 << ", " << j + 1 << "): ";

cin >> matrix1[i][j];

}

}

// Get user input for the elements of the second matrix

cout << "\nEnter the elements of the second 3x3 matrix:" << endl;

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

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

cout << "Enter element at position (" << i + 1 << ", " << j + 1 << "): ";

cin >> matrix2[i][j];

}

}

// Add the matrices

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

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

resultMatrix[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

// Display the result matrix

cout << "\nThe result of the addition of the two matrices is:" << endl;

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

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

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

}

cout << endl;

}

return 0; // Exit the program successfully

}

Output 23 :

PROGRAM 24]

Write a c++ program to implement a student class having name, roll no, rank, address as data members.

code with comments for better understanding

#include <iostream>

#include <string>

using namespace std;

// Define the Student class

class Student {

private:

string name;

int rollNumber;

int rank;

string address;

public:

// Constructor to initialize the data members

Student(string n, int roll, int r, string addr) {

name = n;

rollNumber = roll;

rank = r;

address = addr;

}

// Function to display student details

void displayDetails() {

cout << "Student Details:" << endl;

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

cout << "Roll Number: " << rollNumber << endl;

cout << "Rank: " << rank << endl;

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

}

};

int main() {

// Create an object of the Student class

Student student1("John Doe", 101, 1, "123 Main St, City");

// Display the details of the student

student1.displayDetails();

return 0; // Exit the program successfully

}

Output 24 :

PROGRAM 25]

Write a c++ program to implement sphere class to find surface area and volume.

code with comments for better understanding

#include <iostream>

#include <cmath>

using namespace std;

// Define the Sphere class

class Sphere {

private:

double radius;

public:

// Constructor to initialize the radius

Sphere(double r) {

radius = r;

}

// Function to calculate the surface area of the sphere

double calculateSurfaceArea() const {

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

}

// Function to calculate the volume of the sphere

double calculateVolume() const {

return (4.0 / 3.0) \* M\_PI \* pow(radius, 3);

}

// Function to display sphere details

void displayDetails() const {

cout << "Sphere Details:" << endl;

cout << "Radius: " << radius << endl;

cout << "Surface Area: " << calculateSurfaceArea() << endl;

cout << "Volume: " << calculateVolume() << endl;

}

};

int main() {

// Create an object of the Sphere class

Sphere sphere1(5.0);

// Display the details of the sphere

sphere1.displayDetails();

return 0; // Exit the program successfully

}

Output 25 :

PROGRAM 26]

Write a c++ program to perform following without using library function: A.to reverse the string accepted as argument. B.to count the number of characters in string passes as argument in the form of char array.

code with comments for better understanding

#include <iostream>

using namespace std;

// Function to reverse a string

void reverseString(char str[]) {

// Find the length of the string

int length = 0;

while (str[length] != '\0') {

length++;

}

// Reverse the string

for (int i = 0; i < length / 2; i++) {

char temp = str[i];

str[i] = str[length - i - 1];

str[length - i - 1] = temp;

}

}

// Function to count the number of characters in a string

int countCharacters(char str[]) {

// Initialize count

int count = 0;

// Loop until null character is encountered

for (int i = 0; str[i] != '\0'; i++) {

count++;

}

return count;

}

int main() {

// Part A: Reverse the string

char originalString[] = "Hello";

cout << "Original String: " << originalString << endl;

reverseString(originalString);

cout << "Reversed String: " << originalString << endl;

// Part B: Count the number of characters in the string

char inputString[] = "CountMe";

int charCount = countCharacters(inputString);

cout << "Number of characters in the string: " << charCount << endl;

return 0; // Exit the program successfully

}

Output 26:

PROGRAM 27]

Write a c++ program to perform following without using library function: A.to copy one string into another passed as argument in the form of char array. B.to count the number of consonants, vowels in each word of sentence passed as argument in the form of char array.

code with comments for better understanding

#include <iostream>

using namespace std;

// Function to copy one string into another

void copyString(char source[], char destination[]) {

int i = 0;

while (source[i] != '\0') {

destination[i] = source[i];

i++;

}

destination[i] = '\0'; // Null-terminate the destination string

}

// Function to count consonants and vowels in each word of a sentence

void countConsonantsVowels(char sentence[]) {

int i = 0;

while (sentence[i] != '\0') {

// Skip leading whitespaces

while (sentence[i] == ' ') {

i++;

}

// Count consonants and vowels in the current word

int consonants = 0, vowels = 0;

while (sentence[i] != ' ' && sentence[i] != '\0') {

char currentChar = tolower(sentence[i]); // Convert to lowercase for case-insensitivity

if (currentChar >= 'a' && currentChar <= 'z') {

if (currentChar == 'a' || currentChar == 'e' || currentChar == 'i' || currentChar == 'o' || currentChar == 'u') {

vowels++;

} else {

consonants++;

}

}

i++;

}

// Display the counts for the current word

cout << "Word with consonants and vowels count: " << consonants << ", " << vowels << endl;

}

}

int main() {

// Part A: Copy one string into another

char sourceString[] = "Hello";

char destinationString[10]; // Assuming the destination has enough space

copyString(sourceString, destinationString);

cout << "Source String: " << sourceString << endl;

cout << "Destination String: " << destinationString << endl;

// Part B: Count consonants and vowels in each word of a sentence

char inputSentence[] = "This is a sample sentence.";

cout << "\nInput Sentence: " << inputSentence << endl;

countConsonantsVowels(inputSentence);

return 0; // Exit the program successfully

}

Output 27 :

PROGRAM 28]

Write a c++ program to exchange the content of variable using call by reference.

code with comments for better understanding

#include <iostream>

using namespace std;

// Function to exchange the content of two variables using call by reference

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

int temp = a;

a = b;

b = temp;

}

int main() {

// Declare two variables

int x = 5, y = 10;

// Display the initial values

cout << "Before exchanging: x = " << x << ", y = " << y << endl;

// Call the function to exchange values

exchangeValues(x, y);

// Display the values after exchanging

cout << "After exchanging: x = " << x << ", y = " << y << endl;

return 0; // Exit the program successfully

}

Output 28 :

PROGRAM 29]

Write a c++ program to exchange the content of variable using call by value.

code with comments for better understanding

#include <iostream>

using namespace std;

// Function to exchange the content of two variables using call by value

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

resultA = b;

resultB = a;

}

int main() {

// Declare two variables

int x = 5, y = 10;

// Display the initial values

cout << "Before exchanging: x = " << x << ", y = " << y << endl;

// Call the function to exchange values

int exchangedX, exchangedY;

exchangeValues(x, y, exchangedX, exchangedY);

// Display the values after exchanging

cout << "After exchanging: x = " << exchangedX << ", y = " << exchangedY << endl;

return 0; // Exit the program successfully

}

Output 29 :

PROGRAM 30]

Write a c++ program to sort the integer array.

code with comments for better understanding

#include <iostream>

using namespace std;

// Function to perform Bubble Sort on an integer array

void bubbleSort(int arr[], int size) {

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

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

// Swap if the element found is greater than the next element

if (arr[j] > arr[j + 1]) {

// Perform the swap

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

// Function to display elements of an array

void displayArray(int arr[], int size) {

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

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

}

cout << endl;

}

int main() {

const int size = 5; // Size of the array

int myArray[size];

// Get user input for array elements

cout << "Enter " << size << " integers for the array:" << endl;

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

cin >> myArray[i];

}

// Display the original array

cout << "\nOriginal Array: ";

displayArray(myArray, size);

// Perform Bubble Sort

bubbleSort(myArray, size);

// Display the sorted array

cout << "\nSorted Array: ";

displayArray(myArray, size);

return 0; // Exit the program successfully

}

Output 30 :

PROGRAM 31]

Write a c++ program to take data input from user and sort it in ascending order after sorting using function.

code with comments for better understanding

#include <iostream>

using namespace std;

// Function to perform Bubble Sort on an integer array

void bubbleSort(int arr[], int size) {

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

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

// Swap if the element found is greater than the next element

if (arr[j] > arr[j + 1]) {

// Perform the swap

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

// Function to display elements of an array

void displayArray(int arr[], int size) {

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

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

}

cout << endl;

}

int main() {

int size;

// Get user input for the size of the array

cout << "Enter the size of the array: ";

cin >> size;

int \*myArray = new int[size]; // Dynamic allocation for the array

// Get user input for array elements

cout << "Enter " << size << " integers for the array:" << endl;

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

cin >> myArray[i];

}

// Display the original array

cout << "\nOriginal Array: ";

displayArray(myArray, size);

// Perform Bubble Sort

bubbleSort(myArray, size);

// Display the sorted array

cout << "\nSorted Array (Ascending Order): ";

displayArray(myArray, size);

// Deallocate memory for the array

delete[] myArray;

return 0; // Exit the program successfully

}

Output 31:

PROGRAM 32]

Write a program to input name, address and telephone number of ‘n’ persons (n<=20). Sort according to the name as a primary key and address as the secondary key. Print the sorted telephone directory.

code with comments for better understanding

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

// Structure to represent a person's information

struct Person {

string name;

string address;

string phoneNumber;

};

// Function to compare persons for sorting

bool comparePersons(const Person &a, const Person &b) {

if (a.name != b.name) {

return a.name < b.name; // Sort by name as primary key

}

return a.address < b.address; // Sort by address as secondary key

}

int main() {

const int maxSize = 20; // Maximum number of persons

// Get the number of persons (n)

int n;

cout << "Enter the number of persons (n, max 20): ";

cin >> n;

if (n <= 0 || n > maxSize) {

cout << "Invalid input for the number of persons. Exiting program." << endl;

return 1;

}

// Create a vector to store information about each person

vector<Person> phoneDirectory;

// Input details for each person

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

Person person;

cout << "\nEnter details for person " << i + 1 << ":" << endl;

cout << "Name: ";

cin.ignore(); // Clear the newline character from the buffer

getline(cin, person.name);

cout << "Address: ";

getline(cin, person.address);

cout << "Telephone Number: ";

getline(cin, person.phoneNumber);

phoneDirectory.push\_back(person);

}

// Sort the phone directory based on name and address

sort(phoneDirectory.begin(), phoneDirectory.end(), comparePersons);

// Display the sorted telephone directory

cout << "\nSorted Telephone Directory:" << endl;

for (const auto &person : phoneDirectory) {

cout << "Name: " << person.name << ", Address: " << person.address << ", Telephone: " << person.phoneNumber << endl;

}

return 0; // Exit the program successfully

}

Output 32 :

PROGRAM 33]

Write a c++ program to search the second largest element in an array.

code with comments for better understanding

#include <iostream>

using namespace std;

// Function to find the second largest element in an array

int findSecondLargest(int arr[], int size) {

if (size < 2) {

cout << "Array should have at least two elements." << endl;

return -1; // Return -1 to indicate an error

}

int largest = arr[0];

int secondLargest = arr[0];

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

if (arr[i] > largest) {

secondLargest = largest;

largest = arr[i];

} else if (arr[i] > secondLargest && arr[i] != largest) {

secondLargest = arr[i];

}

}

return secondLargest;

}

int main() {

const int maxSize = 10; // Maximum size of the array

// Get the size of the array from the user

int size;

cout << "Enter the size of the array (max " << maxSize << "): ";

cin >> size;

if (size <= 0 || size > maxSize) {

cout << "Invalid input for the size of the array. Exiting program." << endl;

return 1;

}

// Declare an array of integers

int myArray[maxSize];

// Get array elements from the user

cout << "Enter " << size << " integers for the array:" << endl;

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

cin >> myArray[i];

}

// Find and display the second largest element

int secondLargest = findSecondLargest(myArray, size);

if (secondLargest != -1) {

cout << "The second largest element in the array is: " << secondLargest << endl;

}

return 0; // Exit the program successfully

}

Output 33 :

PROGRAM 34]

Write a c++ program to search the second smallest array in an array.

code with comments for better understanding

#include <iostream>

#include <climits>

using namespace std;

// Function to find the second smallest element in an array

int findSecondSmallest(int arr[], int size) {

if (size < 2) {

cout << "Array should have at least two elements." << endl;

return -1; // Return -1 to indicate an error

}

int smallest = INT\_MAX;

int secondSmallest = INT\_MAX;

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

if (arr[i] < smallest) {

secondSmallest = smallest;

smallest = arr[i];

} else if (arr[i] < secondSmallest && arr[i] != smallest) {

secondSmallest = arr[i];

}

}

return secondSmallest;

}

int main() {

const int maxSize = 10; // Maximum size of the array

// Get the size of the array from the user

int size;

cout << "Enter the size of the array (max " << maxSize << "): ";

cin >> size;

if (size <= 0 || size > maxSize) {

cout << "Invalid input for the size of the array. Exiting program." << endl;

return 1;

}

// Declare an array of integers

int myArray[maxSize];

// Get array elements from the user

cout << "Enter " << size << " integers for the array:" << endl;

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

cin >> myArray[i];

}

// Find and display the second smallest element

int secondSmallest = findSecondSmallest(myArray, size);

if (secondSmallest != -1) {

cout << "The second smallest element in the array is: " << secondSmallest << endl;

}

return 0; // Exit the program successfully

}

Output 34 :

PROGRAM 35]

Write a c++ program to demonstrate static data member.

code with comments for better understanding

#include <iostream>

using namespace std;

// Class with a static data member

class MyClass {

public:

static int count; // Static data member

MyClass() {

count++; // Increment count each time an object is created

}

// Function to get the count

static int getCount() {

return count;

}

};

// Initialize the static data member outside the class definition

int MyClass::count = 0;

int main() {

// Create objects of MyClass

MyClass obj1;

MyClass obj2;

MyClass obj3;

// Display the count using the static member function

cout << "Number of objects created: " << MyClass::getCount() << endl;

return 0; // Exit the program successfully

}

Output 35 :

PROGRAM 36]

Write a c++ program to demonstrate constructor with default argument.

code with comments for better understanding

#include <iostream>

#include <string>

using namespace std;

// Class with a constructor having default arguments

class Person {

private:

string name;

int age;

public:

// Constructor with default arguments

Person(string n = "Unknown", int a = 0) : name(n), age(a) {

// The body of the constructor can be empty or contain additional logic

}

// Function to display person details

void displayDetails() {

cout << "Name: " << name << ", Age: " << age << endl;

}

};

int main() {

// Create objects of Person with different constructors

Person person1("John", 25);

Person person2("Alice");

Person person3; // Uses default values for name and age

// Display details of each person

cout << "Person 1:" << endl;

person1.displayDetails();

cout << "\nPerson 2:" << endl;

person2.displayDetails();

cout << "\nPerson 3:" << endl;

person3.displayDetails();

return 0; // Exit the program successfully

}

Output 36:

PROGRAM 37]

Write a c++ program to demonstrate destructor in inheritance.

code with comments for better understanding

#include <iostream>

#include <string>

using namespace std;

// Base class with a destructor

class Base {

public:

Base() {

cout << "Base class constructor called." << endl;

}

virtual ~Base() {

cout << "Base class destructor called." << endl;

}

};

// Derived class inheriting from Base

class Derived : public Base {

public:

Derived() {

cout << "Derived class constructor called." << endl;

}

~Derived() override {

cout << "Derived class destructor called." << endl;

}

};

int main() {

{

// Create an object of the Derived class

Derived derivedObj;

}

return 0; // Exit the program successfully

}

Output 37:

PROGRAM 38]

Write a c++ program to generate transpose of 3x3 matrix.

code with comments for better understanding

#include <iostream>

using namespace std;

// Function to generate the transpose of a 3x3 matrix

void transposeMatrix(int matrix[3][3], int transpose[3][3]) {

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

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

transpose[j][i] = matrix[i][j];

}

}

}

// Function to display a 3x3 matrix

void displayMatrix(int matrix[3][3]) {

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

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

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

}

cout << endl;

}

}

int main() {

// Declare a 3x3 matrix

int matrix[3][3];

// Get user input for the matrix elements

cout << "Enter the elements of the 3x3 matrix:" << endl;

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

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

cin >> matrix[i][j];

}

}

// Display the original matrix

cout << "\nOriginal Matrix:" << endl;

displayMatrix(matrix);

// Declare a matrix to store the transpose

int transpose[3][3];

// Generate the transpose of the matrix

transposeMatrix(matrix, transpose);

// Display the transposed matrix

cout << "\nTranspose Matrix:" << endl;

displayMatrix(transpose);

return 0; // Exit the program successfully

}

Output 38:

PROGRAM 39]

Write a program having pointer to void to store address of integer variable then print value of integer variable using pointer to void. Perform the same operation for float variable.

code with comments for better understanding

#include <iostream>

using namespace std;

int main() {

// Integer variable

int intValue = 42;

// Float variable

float floatValue = 3.14;

// Pointer to void

void\* ptr;

// Store the address of the integer variable in the pointer to void

ptr = &intValue;

// Print the value of the integer variable using the pointer to void

cout << "Value of integer variable: " << \*(static\_cast<int\*>(ptr)) << endl;

// Store the address of the float variable in the pointer to void

ptr = &floatValue;

// Print the value of the float variable using the pointer to void

cout << "Value of float variable: " << \*(static\_cast<float\*>(ptr)) << endl;

return 0; // Exit the program successfully

}

Output 39 :

PROGRAM 40]

Write a c++ program to create class complex having data members to store real and imaginary part to perform following operation:

A. add 2 complex numbers

B. subtract 2 complex numbers.

code with comments for better understanding

#include <iostream>

using namespace std;

class Complex {

private:

float real;

float imaginary;

public:

// Constructor to initialize complex numbers

Complex(float r = 0.0, float i = 0.0) : real(r), imaginary(i) {}

// Function to add two complex numbers

Complex add(const Complex &c) const {

Complex result;

result.real = real + c.real;

result.imaginary = imaginary + c.imaginary;

return result;

}

// Function to subtract two complex numbers

Complex subtract(const Complex &c) const {

Complex result;

result.real = real - c.real;

result.imaginary = imaginary - c.imaginary;

return result;

}

// Function to display the complex number

void display() const {

cout << real << " + " << imaginary << "i";

}

};

int main() {

// Create two complex numbers

Complex complex1(3.0, 4.0);

Complex complex2(1.5, 2.5);

// Add the complex numbers

Complex sum = complex1.add(complex2);

// Subtract the complex numbers

Complex difference = complex1.subtract(complex2);

// Display the results

cout << "Complex Number 1: ";

complex1.display();

cout << endl;

cout << "Complex Number 2: ";

complex2.display();

cout << endl;

cout << "\nSum of Complex Numbers: ";

sum.display();

cout << endl;

cout << "Difference of Complex Numbers: ";

difference.display();

cout << endl;

return 0; // Exit the program successfully

}

Output 40 :

PROGRAM 41]

Create class polar having data member radius and angle it contains member function for taking the input in data member and member function for displaying values of data members. It contains declaration of friend function which accept 2 objects of class polar and returns object of class polar after addition.

code with comments for better understanding

#include <iostream>

#include <cmath>

using namespace std;

class Polar;

class Polar {

private:

double radius;

double angle;

public:

// Function to take input for radius and angle

void getInput() {

cout << "Enter radius: ";

cin >> radius;

cout << "Enter angle in degrees: ";

cin >> angle;

}

// Function to display values of data members

void display() const {

cout << "Radius: " << radius << ", Angle: " << angle << " degrees";

}

// Friend function to add two Polar objects

friend Polar addPolar(const Polar &p1, const Polar &p2);

};

// Function definition for friend function to add two Polar objects

Polar addPolar(const Polar &p1, const Polar &p2) {

Polar result;

result.radius = sqrt(pow(p1.radius, 2) + pow(p2.radius, 2) + 2 \* p1.radius \* p2.radius \* cos(p1.angle - p2.angle));

result.angle = p1.angle + p2.angle;

return result;

}

int main() {

// Create two Polar objects

Polar polar1, polar2;

// Take input for the first Polar object

cout << "Enter details for Polar Object 1:" << endl;

polar1.getInput();

// Take input for the second Polar object

cout << "\nEnter details for Polar Object 2:" << endl;

polar2.getInput();

// Display values of the two Polar objects

cout << "\nPolar Object 1: ";

polar1.display();

cout << endl;

cout << "Polar Object 2: ";

polar2.display();

cout << endl;

// Add the two Polar objects using the friend function

Polar sum = addPolar(polar1, polar2);

// Display the result of addition

cout << "\nResult after addition:" << endl;

sum.display();

cout << endl;

return 0; // Exit the program successfully

}

Output 41:

PROGRAM 42]

Create class polar having data member radius and angle it contains member function for taking the input in data member and member function for displaying values of data members. It contains declaration of friend function which accept 2 objects of class polar and returns object of class polar after addition.

code with comments for better understanding

#include <iostream>

#include <cmath>

using namespace std;

class Polar {

private:

double radius;

double angle;

public:

// Function to take input for radius and angle

void getInput() {

cout << "Enter radius: ";

cin >> radius;

cout << "Enter angle in degrees: ";

cin >> angle;

}

// Function to display values of data members

void display() const {

cout << "Radius: " << radius << ", Angle: " << angle << " degrees";

}

// Friend function declaration to add two Polar objects

friend Polar addPolar(const Polar &p1, const Polar &p2);

};

// Friend function definition to add two Polar objects

Polar addPolar(const Polar &p1, const Polar &p2) {

Polar result;

result.radius = sqrt(pow(p1.radius, 2) + pow(p2.radius, 2) + 2 \* p1.radius \* p2.radius \* cos(p1.angle - p2.angle));

result.angle = p1.angle + p2.angle;

return result;

}

int main() {

// Create two Polar objects

Polar polar1, polar2;

// Take input for the first Polar object

cout << "Enter details for Polar Object 1:" << endl;

polar1.getInput();

// Take input for the second Polar object

cout << "\nEnter details for Polar Object 2:" << endl;

polar2.getInput();

// Display values of the two Polar objects

cout << "\nPolar Object 1: ";

polar1.display();

cout << endl;

cout << "Polar Object 2: ";

polar2.display();

cout << endl;

// Add the two Polar objects using the friend function

Polar sum = addPolar(polar1, polar2);

// Display the result of addition

cout << "\nResult after addition:" << endl;

sum.display();

cout << endl;

return 0; // Exit the program successfully

}

Output 42 :

PROGRAM 43]

Write a c++ program to read data on an employee and compute the net salary of each employee DA=42 % of basic and income tax=30% of the gross salary.

code with comments for better understanding

#include <iostream>

#include <string>

using namespace std;

class Employee {

private:

string name;

double basicSalary;

public:

// Function to read data for an employee

void readData() {

cout << "Enter employee name: ";

getline(cin, name);

cout << "Enter basic salary: ";

cin >> basicSalary;

}

// Function to compute net salary

double computeNetSalary() const {

double da = 0.42 \* basicSalary; // DA is 42% of basic salary

double grossSalary = basicSalary + da;

double incomeTax = 0.30 \* grossSalary; // Income Tax is 30% of gross salary

double netSalary = grossSalary - incomeTax;

return netSalary;

}

// Function to display employee details and net salary

void displayDetails() const {

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

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

cout << "Basic Salary: " << basicSalary << endl;

double netSalary = computeNetSalary();

cout << "Net Salary: " << netSalary << endl;

}

};

int main() {

// Create an object of the Employee class

Employee employee;

// Read data for the employee

cout << "Enter details for the employee:" << endl;

employee.readData();

// Display employee details and net salary

employee.displayDetails();

return 0; // Exit the program successfully

}

Output 43 :

PROGRAM 44]

Write a c++ program to evaluate the sum of series 1+2+….n using while loop.

code with comments for better understanding

#include <iostream>

using namespace std;

int main() {

// Declare variables

int n, i = 1, sum = 0;

// Get the value of n from the user

cout << "Enter the value of n: ";

cin >> n;

// Use a while loop to calculate the sum of the series

while (i <= n) {

sum += i;

i++;

}

// Display the sum of the series

cout << "Sum of the series 1 + 2 + ... + " << n << " is: " << sum << endl;

return 0; // Exit the program successfully

}

Output 44 :

PROGRAM 45]

Write a c++ program to find roots of quadratic equation.

code with comments for better understanding

#include <iostream>

#include <cmath>

using namespace std;

int main() {

// Declare variables for coefficients

double a, b, c;

// Get coefficients from the user

cout << "Enter the coefficients of the quadratic equation (a, b, c): ";

cin >> a >> b >> c;

// Calculate the discriminant

double discriminant = b \* b - 4 \* a \* c;

// Check the nature of roots based on the discriminant

if (discriminant > 0) {

// Two real and distinct roots

double root1 = (-b + sqrt(discriminant)) / (2 \* a);

double root2 = (-b - sqrt(discriminant)) / (2 \* a);

cout << "Root 1 = " << root1 << endl;

cout << "Root 2 = " << root2 << endl;

} else if (discriminant == 0) {

// One real and repeated root

double root = -b / (2 \* a);

cout << "Root = " << root << endl;

} else {

// Complex roots

double realPart = -b / (2 \* a);

double imaginaryPart = sqrt(-discriminant) / (2 \* a);

cout << "Root 1 = " << realPart << " + " << imaginaryPart << "i" << endl;

cout << "Root 2 = " << realPart << " - " << imaginaryPart << "i" << endl;

}

return 0; // Exit the program successfully

}

Output 45 :

PROGRAM 46]

Write a C++ program to create class mother having data member to store salary of mother, create another class father to store salary of father. use friend function which accept object of father and mother and print salary of mother and father.

code with comments for better understanding

#include <iostream>

using namespace std;

// Forward declaration of the Father class

class Father;

// Mother class with a friend function

class Mother {

private:

int salaryMother;

public:

// Constructor to initialize salary of mother

Mother(int salary) : salaryMother(salary) {}

// Friend function declaration to access Father's salary

friend void displaySalaries(const Mother& mother, const Father& father);

};

// Father class

class Father {

private:

int salaryFather;

public:

// Constructor to initialize salary of father

Father(int salary) : salaryFather(salary) {}

// Friend function declaration to access Mother's salary

friend void displaySalaries(const Mother& mother, const Father& father);

};

// Friend function definition to display salaries of Mother and Father

void displaySalaries(const Mother& mother, const Father& father) {

cout << "Mother's Salary: " << mother.salaryMother << endl;

cout << "Father's Salary: " << father.salaryFather << endl;

}

int main() {

// Create objects of Mother and Father classes

Mother mother(50000);

Father father(60000);

// Call the friend function to display salaries

displaySalaries(mother, father);

return 0; // Exit the program successfully

}

Output 46 :

PROGRAM 47]

Write a C++ program using class which uses static overloaded function to swap 2 integers,2 float method using reference variable.

code with comments for better understanding

#include <iostream>

using namespace std;

class Swapper {

public:

// Static function to swap two integers

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

int temp = a;

a = b;

b = temp;

}

// Static function to swap two floats

static void swap(float &x, float &y) {

float temp = x;

x = y;

y = temp;

}

};

int main() {

// Swap two integers

int int1 = 5, int2 = 10;

cout << "Before swapping integers: " << int1 << " " << int2 << endl;

Swapper::swap(int1, int2);

cout << "After swapping integers: " << int1 << " " << int2 << endl;

// Swap two floats

float float1 = 3.14, float2 = 2.718;

cout << "\nBefore swapping floats: " << float1 << " " << float2 << endl;

Swapper::swap(float1, float2);

cout << "After swapping floats: " << float1 << " " << float2 << endl;

return 0; // Exit the program successfully

}

Output 47 :

PROGRAM 48]

Write a C++ program using class which uses static overloaded function to swap 2 integers,2 float method using passing by address.

code with comments for better understanding

#include <iostream>

using namespace std;

class Swapper {

public:

// Static function to swap two integers using pointers

static void swap(int\* a, int\* b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

// Static function to swap two floats using pointers

static void swap(float\* x, float\* y) {

float temp = \*x;

\*x = \*y;

\*y = temp;

}

};

int main() {

// Swap two integers using pointers

int int1 = 5, int2 = 10;

cout << "Before swapping integers: " << int1 << " " << int2 << endl;

Swapper::swap(&int1, &int2);

cout << "After swapping integers: " << int1 << " " << int2 << endl;

// Swap two floats using pointers

float float1 = 3.14, float2 = 2.718;

cout << "\nBefore swapping floats: " << float1 << " " << float2 << endl;

Swapper::swap(&float1, &float2);

cout << "After swapping floats: " << float1 << " " << float2 << endl;

return 0; // Exit the program successfully

}

Output 48:

PROGRAM 49]

Write a c++ program to create class string having pointer to char as data member and provide following facilities:

a. overloaded operators to ‘+’ to add two string object.

b. overloaded operator ‘=’ to assign string object to another string object.

code with comments for better understanding

#include <iostream>

#include <cstring>

using namespace std;

class String {

private:

char\* str;

public:

// Constructor to initialize the string

String(const char\* s = nullptr) {

if (s) {

str = new char[strlen(s) + 1];

strcpy(str, s);

} else {

str = nullptr;

}

}

// Destructor to free the dynamically allocated memory

~String() {

delete[] str;

}

// Overloaded '+' operator to add two string objects

String operator+(const String& other) const {

char\* result = new char[strlen(str) + strlen(other.str) + 1];

strcpy(result, str);

strcat(result, other.str);

return String(result);

}

// Overloaded '=' operator to assign one string object to another

String& operator=(const String& other) {

if (this != &other) {

delete[] str;

str = new char[strlen(other.str) + 1];

strcpy(str, other.str);

}

return \*this;

}

// Function to display the string

void display() const {

cout << "String: " << (str ? str : "Empty") << endl;

}

};

int main() {

// Create two String objects

String str1("Hello"), str2("World");

// Display the original strings

cout << "Original Strings:" << endl;

str1.display();

str2.display();

// Use the '+' operator to add two strings

String result = str1 + str2;

cout << "\nAfter Adding Strings:" << endl;

result.display();

// Use the '=' operator to assign one string to another

String assignedString;

assignedString = result;

cout << "\nAfter Assigning String:" << endl;

assignedString.display();

return 0; // Exit the program successfully

}

Output 49 :

PROGRAM 50]

Write a c++ program to find the biggest of 3 number using pointer and function.

code with comments for better understanding

#include <iostream>

using namespace std;

// Function to find the biggest of three numbers using pointers

int findBiggest(int\* num1, int\* num2, int\* num3) {

if (\*num1 >= \*num2 && \*num1 >= \*num3) {

return \*num1;

} else if (\*num2 >= \*num1 && \*num2 >= \*num3) {

return \*num2;

} else {

return \*num3;

}

}

int main() {

// Declare variables to store three numbers

int num1, num2, num3;

// Get input from the user

cout << "Enter three numbers:" << endl;

cin >> num1 >> num2 >> num3;

// Call the function to find the biggest number using pointers

int biggest = findBiggest(&num1, &num2, &num3);

// Display the result

cout << "The biggest number among " << num1 << ", " << num2 << ", and " << num3 << " is: " << biggest << endl;

return 0; // Exit the program successfully

}

Output 50 ;

PROGRAM 51]

Write a c++ program to create inline function to calculate area of circle.

code with comments for better understanding

#include <iostream>

using namespace std;

// Inline function to calculate the area of a circle

inline double calculateCircleArea(double radius) {

return 3.14159 \* radius \* radius;

}

int main() {

// Declare variable to store the radius of the circle

double radius;

// Get input from the user

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

cin >> radius;

// Call the inline function to calculate the area of the circle

double area = calculateCircleArea(radius);

// Display the result

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

return 0; // Exit the program successfully

}

Output 51:

PROGRAM 52]

Write a c++ program to create inline function to take 2 arguments and should return the minimum values.

code with comments for better understanding

#include <iostream>

using namespace std;

// Inline function to find the minimum of two values

inline int findMinimum(int a, int b) {

return (a < b) ? a : b;

}

int main() {

// Declare variables to store two values

int num1, num2;

// Get input from the user

cout << "Enter two numbers: ";

cin >> num1 >> num2;

// Call the inline function to find the minimum of the two numbers

int minimum = findMinimum(num1, num2);

// Display the result

cout << "The minimum of " << num1 << " and " << num2 << " is: " << minimum << endl;

return 0; // Exit the program successfully

}

Output 52:

PROGRAM 53]

Write a c++ program to calculate factorial of number using copy constructor.

code with comments for better understanding

#include <iostream>

using namespace std;

class FactorialCalculator {

private:

int number;

unsigned long long result; // To handle larger factorials

public:

// Parameterized constructor

FactorialCalculator(int num) : number(num), result(1) {}

// Copy constructor

FactorialCalculator(const FactorialCalculator& other) : number(other.number), result(other.result) {}

// Function to calculate factorial

void calculateFactorial() {

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

result \*= i;

}

}

// Function to display the result

void displayResult() const {

cout << "Factorial of " << number << " is: " << result << endl;

}

};

int main() {

// Get input from the user

int inputNumber;

cout << "Enter a number to calculate factorial: ";

cin >> inputNumber;

// Create an object of FactorialCalculator

FactorialCalculator calculator(inputNumber);

// Use copy constructor to create another object

FactorialCalculator copyCalculator = calculator;

// Calculate factorial for the original object

calculator.calculateFactorial();

// Display the result for the original object

calculator.displayResult();

// Display the result for the object created using the copy constructor

copyCalculator.displayResult();

return 0; // Exit the program successfully

}

Output 53 :

PROGRAM 54]

Write a c++ program to copy content of one file into another after removing extra spaces between word and name of file should come from command line argument.

code with comments for better understanding

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

// Function to remove extra spaces between words

string removeExtraSpaces(const string& input) {

string result;

bool spaceFound = false;

for (char ch : input) {

if (ch == ' ') {

if (!spaceFound) {

result += ch;

spaceFound = true;

}

} else {

result += ch;

spaceFound = false;

}

}

return result;

}

// Function to copy content from one file to another after removing extra spaces

void copyFileWithSpacesRemoval(const string& inputFileName, const string& outputFileName) {

ifstream inputFile(inputFileName);

if (!inputFile.is\_open()) {

cerr << "Error opening input file." << endl;

return;

}

ofstream outputFile(outputFileName);

if (!outputFile.is\_open()) {

cerr << "Error opening output file." << endl;

inputFile.close();

return;

}

string line;

while (getline(inputFile, line)) {

string processedLine = removeExtraSpaces(line);

outputFile << processedLine << '\n';

}

cout << "File content copied with extra spaces removed successfully." << endl;

inputFile.close();

outputFile.close();

}

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

// Check if command-line arguments are provided

if (argc != 3) {

cerr << "Usage: " << argv[0] << " <inputFileName> <outputFileName>" << endl;

return 1; // Exit with an error code

}

// Get input and output file names from command-line arguments

string inputFileName(argv[1]);

string outputFileName(argv[2]);

// Call the function to copy content and remove extra spaces

copyFileWithSpacesRemoval(inputFileName, outputFileName);

return 0; // Exit the program successfully

}

Output 54 :

PROGRAM 55]

Write a c++ program to demonstrate virtual function.

code with comments for better understanding

#include <iostream>

using namespace std;

// Base class with a virtual function

class Shape {

public:

// Virtual function to calculate the area

virtual double calculateArea() const {

return 0.0;

}

// Virtual function to display the type of shape

virtual void displayType() const {

cout << "This is a generic shape." << endl;

}

};

// Derived class 1: Circle

class Circle : public Shape {

private:

double radius;

public:

// Constructor

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

// Override the virtual function to calculate the area for a circle

double calculateArea() const override {

return 3.14159 \* radius \* radius;

}

// Override the virtual function to display the type of shape

void displayType() const override {

cout << "This is a circle." << endl;

}

};

// Derived class 2: Rectangle

class Rectangle : public Shape {

private:

double length;

double width;

public:

// Constructor

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

// Override the virtual function to calculate the area for a rectangle

double calculateArea() const override {

return length \* width;

}

// Override the virtual function to display the type of shape

void displayType() const override {

cout << "This is a rectangle." << endl;

}

};

int main() {

// Create objects of Shape, Circle, and Rectangle

Shape\* shape = new Shape();

Shape\* circle = new Circle(5.0);

Shape\* rectangle = new Rectangle(4.0, 6.0);

// Call virtual functions and display the type of shapes

cout << "Area of the shape: " << shape->calculateArea() << endl;

shape->displayType();

cout << "\nArea of the circle: " << circle->calculateArea() << endl;

circle->displayType();

cout << "\nArea of the rectangle: " << rectangle->calculateArea() << endl;

rectangle->displayType();

// Remember to delete dynamically allocated objects

delete shape;

delete circle;

delete rectangle;

return 0; // Exit the program successfully

}

PROGRAM 56]

Write a C++ program to demonstrate pure virtual function.

code with comments for better understanding

#include <iostream>

using namespace std;

// Abstract base class with a pure virtual function

class Shape {

public:

// Pure virtual function to calculate the area (abstract method)

virtual double calculateArea() const = 0;

// Virtual function to display the type of shape

virtual void displayType() const {

cout << "This is a generic shape." << endl;

}

// Destructor

virtual ~Shape() {}

};

// Derived class 1: Circle

class Circle : public Shape {

private:

double radius;

public:

// Constructor

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

// Override the pure virtual function to calculate the area for a circle

double calculateArea() const override {

return 3.14159 \* radius \* radius;

}

// Override the virtual function to display the type of shape

void displayType() const override {

cout << "This is a circle." << endl;

}

};

// Derived class 2: Rectangle

class Rectangle : public Shape {

private:

double length;

double width;

public:

// Constructor

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

// Override the pure virtual function to calculate the area for a rectangle

double calculateArea() const override {

return length \* width;

}

// Override the virtual function to display the type of shape

void displayType() const override {

cout << "This is a rectangle." << endl;

}

};

int main() {

// Create objects of Circle and Rectangle

Shape\* circle = new Circle(5.0);

Shape\* rectangle = new Rectangle(4.0, 6.0);

// Call pure virtual functions and display the type of shapes

cout << "Area of the circle: " << circle->calculateArea() << endl;

circle->displayType();

cout << "\nArea of the rectangle: " << rectangle->calculateArea() << endl;

rectangle->displayType();

// Remember to delete dynamically allocated objects

delete circle;

delete rectangle;

return 0; // Exit the program successfully

}

Output 56 :