16. Write a Matrix class with attributes: row, col, and dynamically created two-dimensional int array. Define constructors to allocate memory dynamically for the dimensions supplied. Write methods to set, get values of individual matrix elements, display contents of the matrix, etc.

#include <iostream>

using namespace std;

class Matrix {

private:

    int row;

    int col;

    int\*\* matrix;

public:

    // Constructor to allocate memory for the matrix

    Matrix(int numRows, int numCols) : row(numRows), col(numCols) {

        // Allocate memory for the matrix

        matrix = new int\*[row];

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

            matrix[i] = new int[col];

        }

    }

    // Destructor to free allocated memory

    ~Matrix() {

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

            delete[] matrix[i];

        }

        delete[] matrix;

    }

    // Method to set the value of an individual element in the matrix

    void setElement(int i, int j, int value) {

        if (i >= 0 && i < row && j >= 0 && j < col) {

            matrix[i][j] = value;

        } else {

            cout << "Invalid indices." << endl;

        }

    }

    // Method to get the value of an individual element in the matrix

    int getElement(int i, int j) const {

        if (i >= 0 && i < row && j >= 0 && j < col) {

            return matrix[i][j];

        } else {

            cout << "Invalid indices." << endl;

            return 0; // Return a default value if indices are invalid

        }

    }

    // Method to display the contents of the matrix

    void displayMatrix() const {

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

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

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

            }

            cout << std::endl;

        }

    }

};

int main() {

    // Example usage of the Matrix class

    int numRows, numCols;

    // User input for matrix dimensions

    cout << "Enter the number of rows: ";

    cin >> numRows;

    cout << "Enter the number of columns: ";

    cin >> numCols;

    // Create a Matrix object with the specified dimensions

    Matrix myMatrix(numRows, numCols);

    // Set and display matrix elements

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

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

            int value;

            cout << "Enter value for matrix element at position (" << i << ", " << j << "): ";

            cin >> value;

            myMatrix.setElement(i, j, value);

        }

    }

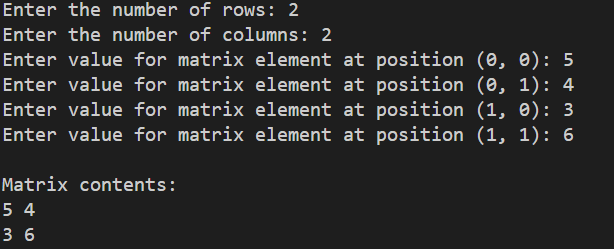
    cout << "\nMatrix contents:\n";

    myMatrix.displayMatrix();

    return 0;

}

Output:



17. Write a function that receives two matrix objects as arguments and returns a new matrix object containing their multiplication result.

#include <iostream>

using namespace std;

class Matrix {

private:

    int row;

    int col;

    int\*\* matrix;

public:

    // Constructor to allocate memory for the matrix

    Matrix(int numRows, int numCols) : row(numRows), col(numCols) {

        // Allocate memory for the matrix

        matrix = new int\*[row];

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

            matrix[i] = new int[col];

        }

    }

    // Destructor to free allocated memory

    ~Matrix() {

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

            delete[] matrix[i];

        }

        delete[] matrix;

    }

    // Method to set the value of an individual element in the matrix

    void setElement(int i, int j, int value) {

        if (i >= 0 && i < row && j >= 0 && j < col) {

            matrix[i][j] = value;

        } else {

            cout << "Invalid indices." << endl;

        }

    }

    // Method to get the value of an individual element in the matrix

    int getElement(int i, int j) const {

        if (i >= 0 && i < row && j >= 0 && j < col) {

            return matrix[i][j];

        } else {

            cout << "Invalid indices." << endl;

            return 0; // Return a default value if indices are invalid

        }

    }

    // Method to display the contents of the matrix

    void displayMatrix() const {

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

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

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

            }

            cout << std::endl;

        }

    }

    static Matrix multiply(const Matrix& mat1, const Matrix& mat2) {

        // Check if multiplication is possible

        if (mat1.col != mat2.row) {

            cerr << "Matrix multiplication is not possible. Invalid dimensions." << endl;

            return Matrix(0, 0);  // Return an empty matrix

        }

        // Create a result matrix with dimensions mat1.row x mat2.col

        static Matrix result(mat1.row, mat2.col);

        // Perform matrix multiplication

        for (int i = 0; i < mat1.row; i++) {

            for (int j = 0; j < mat2.col; j++) {

                int sum = 0;

                for (int k = 0; k < mat1.col; k++) {

                    sum += mat1.matrix[i][k] \* mat2.matrix[k][j];

                }

                result.setElement(i, j, sum);

            }

        }

        return result;

    }

};

int main() {

    // Example usage of the Matrix class and multiplication function

    int numRows1, numCols1, numRows2, numCols2;

    numRows1= numCols1= numRows2= numCols2= 2;

    // Create two Matrix objects with the specified dimensions

    Matrix mat1(numRows1, numCols1);

    Matrix mat2(numRows2, numCols2);

    // Set matrix elements (for simplicity, elements are set manually here)

    mat1.setElement(0, 0, 1); mat1.setElement(0, 1, 2);

    mat1.setElement(1, 0, 3); mat1.setElement(1, 1, 4);

    mat2.setElement(0, 0, 5); mat2.setElement(0, 1, 6);

    mat2.setElement(1, 0, 7); mat2.setElement(1, 1, 8);

    // Display matrix contents

    cout << "\nMatrix 1:\n";

    mat1.displayMatrix();

    cout << "\nMatrix 2:\n";

    mat2.displayMatrix();

    // Multiply matrices and display the result

    Matrix result = Matrix::multiply(mat1, mat2);

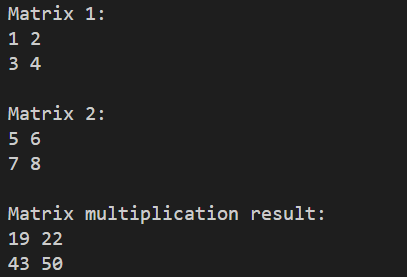
    cout << "\nMatrix multiplication result:\n";

    result.displayMatrix();

    return 0;

}

Output:



18. A bookshop maintains the inventory of books that are being sold at the shop. The list includes details such as author, title, price, publisher, and stock position. Whenever a customer wants a book, the salesperson inputs the title and author and the system searches the list and displays whether it is available or not. If it is not, an appropriate message is displayed. If it is, then the system displays the book details and requests for the number of copies required. If the requested copies are available, the total cost of the requested copies is displayed; otherwise “Required copies not in stock” is displayed. Design a system using a class called books with suitable member functions and constructors. Use new operator in constructors to allocate memory space required.

#include <iostream>

#include <cstring>

class Book {

private:

    char\* title;

    char\* author;

    double price;

    char\* publisher;

    int stockPosition;

public:

    // Constructor to initialize book details

    Book(const char\* t, const char\* a, double p, const char\* pub, int stock)

        : price(p), stockPosition(stock) {

        // Allocate memory for title, author, and publisher

        title = new char[strlen(t) + 1];

        author = new char[strlen(a) + 1];

        publisher = new char[strlen(pub) + 1];

        // Copy values to allocated memory

        strcpy(title, t);

        strcpy(author, a);

        strcpy(publisher, pub);

    }

    // Destructor to free allocated memory

    ~Book() {

        delete[] title;

        delete[] author;

        delete[] publisher;

    }

    // Member function to check availability and display details

    void checkAvailability(const char\* inputTitle, const char\* inputAuthor, int numCopies) const {

        if (strcmp(title, inputTitle) == 0 && strcmp(author, inputAuthor) == 0) {

            // Book found

            if (stockPosition >= numCopies) {

                // Sufficient copies in stock

                std::cout << "Book Details:\n";

                std::cout << "Title: " << title << "\nAuthor: " << author << "\nPrice: $" << price

                          << "\nPublisher: " << publisher << "\nIn Stock: " << stockPosition << " copies\n";

                // Display total cost

                double totalCost = price \* numCopies;

                std::cout << "Total Cost for " << numCopies << " copies: $" << totalCost << std::endl;

            } else {

                // Insufficient copies in stock

                std::cout << "Required copies not in stock.\n";

            }

        } else {

            // Book not found

            std::cout << "Book not available.\n";

        }

    }

};

int main() {

    // Example usage of the Book class

    Book book1("The Catcher in the Rye", "J.D. Salinger", 15.99, "Little, Brown and Company", 10);

    Book book2("To Kill a Mockingbird", "Harper Lee", 12.50, "J.B. Lippincott & Co.", 15);

    // Check availability and display details

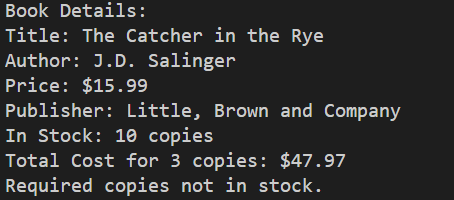
    book1.checkAvailability("The Catcher in the Rye", "J.D. Salinger", 3);

    book2.checkAvailability("To Kill a Mockingbird", "Harper Lee", 20);

    return 0;

}

Output:



19. WAP to create a structure named company which has name, phone, and no\_of\_employee as member variables. Create other structure named address and store address of company like plot no, city, state including pin name. Read name of company, its address, phone, and no\_of\_employee for at least 3 companies using an array. Display data of each company. Make appropriate assumptions to run the program.

// WAP to create a structure named company which has name, phone, and no\_of\_employee as

// member variables. Create other structure named address and store address of company like

// plot no, city, state including pin name. Read name of company, its address, phone, and

// no\_of\_employee for at least 3 companies using an array. Display data of each company.

// Make appropriate assumptions to run the program.

#include <stdio.h>

// Structure for company information

struct address {

    char plotNo[50];

    char city[50];

    char state[50];

    char pin[10];

};

// Structure for company information

struct company {

    char name[50];

    char phone[15];

    int no\_of\_employees;

    struct address company\_address;  // Nested structure for address

};

int main() {

    // Assume a maximum of 3 companies

    struct company companies[3];

    // Input data for each company

    for (int i = 0; i < 3; ++i) {

        printf("Enter details for Company %d:\n", i + 1);

        // Company name

        printf("Name: ");

        scanf("%s", companies[i].name);

        // Company phone

        printf("Phone: ");

        scanf("%s", companies[i].phone);

        // Number of employees

        printf("Number of Employees: ");

        scanf("%d", &companies[i].no\_of\_employees);

        // Company address

        printf("Address:\n");

        printf("Plot No: ");

        scanf("%s", companies[i].company\_address.plotNo);

        printf("City: ");

        scanf("%s", companies[i].company\_address.city);

        printf("State: ");

        scanf("%s", companies[i].company\_address.state);

        printf("PIN: ");

        scanf("%s", companies[i].company\_address.pin);

    }

    // Display data for each company

    printf("\nCompany Details:\n");

    for (int i = 0; i < 3; ++i) {

        printf("Company %d:\n", i + 1);

        printf("Name: %s\n", companies[i].name);

        printf("Phone: %s\n", companies[i].phone);

        printf("Number of Employees: %d\n", companies[i].no\_of\_employees);

        printf("Address: %s, %s, %s - %s\n",

               companies[i].company\_address.plotNo,

               companies[i].company\_address.city,

               companies[i].company\_address.state,

               companies[i].company\_address.pin);

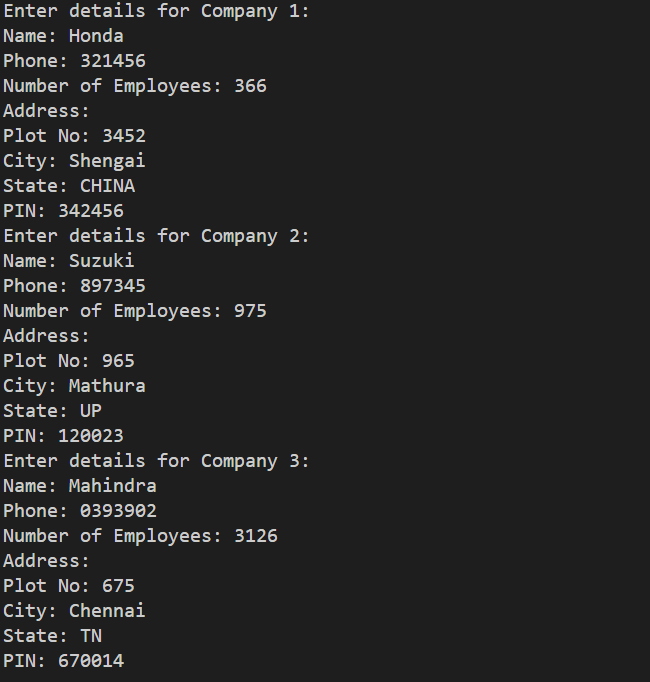
        printf("\n");

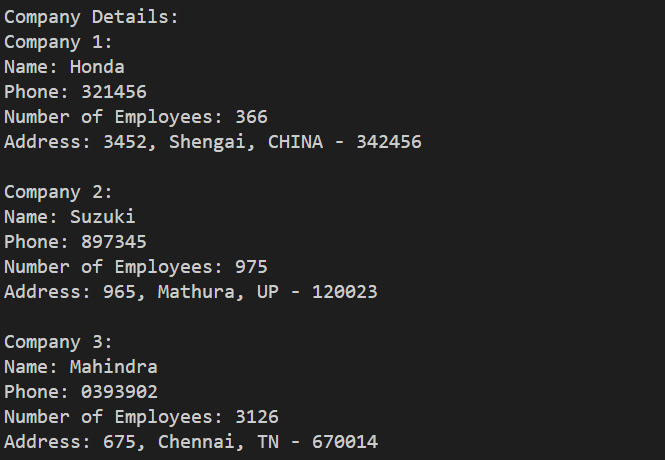
    }

    return 0;

}

Output:





20. WAP to compute Sin(x) using Taylor series approximation.



//WAP to compute Sin(x) using Taylor series approximation.

#include <stdio.h>

#include <math.h>

double calculate\_sin(double x, int terms) {

    double result = 0;

    int i;

    for (i = 0; i < terms; i++) {

        int power = 2 \* i + 1;

        double term = pow(-1, i) \* pow(x, power) / tgamma(power + 1);

        result += term;

    }

    return result;

}

int main() {

    double x;

    int terms;

    printf("Enter the value of x in radians: ");

    scanf("%lf", &x);

    printf("Enter the number of terms in the Taylor series: ");

    scanf("%d", &terms);

    double sin\_x = calculate\_sin(x, terms);

    printf("sin(%lf) = %lf\n", x, sin\_x);

    return 0;

}

Output:

