Assignment 16

Multi dimensional array

#include <stdio.h>

// Function to read a 3x3 matrix from the user

void readMatrix(int matrix[3][3])

{

printf("Enter the elements of the matrix (3x3):\n");

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

{

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

{

scanf("%d", &matrix[i][j]);

}

}

}

// Function to calculate the sum of two 3x3 matrices

void sumMatrices(int matrix1[3][3], int matrix2[3][3], int result[3][3])

{

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

{

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

{

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

}

}

}

// Function to calculate the product of two 3x3 matrices

void multiplyMatrices(int matrix1[3][3], int matrix2[3][3], int result[3][3])

{

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

{

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

{

result[i][j] = 0;

for (int k = 0; k < 3; k++)

{

result[i][j] += matrix1[i][k] \* matrix2[k][j];

}

}

}

}

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

void transposeMatrix(int matrix[3][3], int result[3][3])

{

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

{

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

{

result[i][j] = matrix[j][i];

}

}

}

// Function to find the sum of right diagonals of a 3x3 matrix

int sumRightDiagonals(int matrix[3][3])

{

int sum = 0;

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

{

sum += matrix[i][2 - i];

}

return sum;

}

// Function to find the sum of rows and columns of a 3x3 matrix

void sumRowsAndColumns(int matrix[3][3])

{

int rowSum[3] = {0};

int colSum[3] = {0};

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

{

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

{

rowSum[i] += matrix[i][j];

colSum[j] += matrix[i][j];

}

}

printf("Sum of rows: ");

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

{

printf("%d ", rowSum[i]);

}

printf("\n\n");

printf("Sum of columns: ");

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

{

printf("%d ", colSum[j]);

}

printf("\n\n");

}

// Function to print the lower triangular of a 3x3 matrix

void printLowerTriangular(int matrix[3][3])

{

printf("Lower Triangular Matrix:\n");

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

{

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

{

if (i >= j)

{

printf("%d ", matrix[i][j]);

}

else

{

printf("0 ");

}

}

printf("\n");

}

}

// Function to print the upper triangular of a 3x3 matrix

void printUpperTriangular(int matrix[3][3])

{

printf("Upper Triangular Matrix:\n");

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

{

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

{

if (i <= j) {

printf("%d ", matrix[i][j]);

}

else {

printf("0 ");

}

}

printf("\n");

}

}

// Function to determine whether a matrix is sparse

int isSparseMatrix(int mat[3][3])

{

int countZeros = 0;

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

{

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

{

if (mat[i][j] == 0)

{

countZeros++;

}

}

}

return (countZeros > (3 \* 3) / 2);

}

// Function to find the row with the maximum number of 1s

int findRowWithMaxOnes(int mat[3][3])

{

int maxOnes = 0;

int rowWithMaxOnes = -1;

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

{

int onesInRow = 0;

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

{

if (mat[i][j] == 1){

onesInRow++;

}

}

if (onesInRow > maxOnes)

{

maxOnes = onesInRow;

rowWithMaxOnes = i;

}

}

return rowWithMaxOnes;

}

// Main

int main()

{

int matrix1[3][3], matrix2[3][3], result[3][3], transpose[3][3];

printf("Matrix 1:\n");

readMatrix(matrix1);

printf("Matrix 2:\n");

readMatrix(matrix2);

printf("\n");

sumMatrices(matrix1, matrix2, result);

printf("Sum of two matrices:\n");

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

{

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

{

printf("%d ", result[i][j]);

}

printf("\n");

}

printf("\n");

multiplyMatrices(matrix1, matrix2, result);

printf("Product of two matrices:\n");

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

{

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

{

printf("%d ", result[i][j]);

}

printf("\n");

}

printf("\n");

transposeMatrix(matrix1, transpose);

printf("Transpose of the matrix:\n");

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

{

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

{

printf("%d ", transpose[i][j]);

}

printf("\n");

}

printf("\n");

printf("Sum of right diagonals: %d\n\n", sumRightDiagonals(matrix1));

sumRowsAndColumns(matrix1);

printLowerTriangular(matrix1);

printUpperTriangular(matrix1);

if (isSparseMatrix(matrix1)){

printf("\nThe matrix is sparse.\n\n");

}

else{

printf("\nThe matrix is not sparse.\n\n");

}

int rowWithMaxOnes = findRowWithMaxOnes(matrix1);

if (rowWithMaxOnes != -1){

printf("Row with the maximum number of 1s: %d\n\n", rowWithMaxOnes + 1);

}

else{

printf("No row with 1s found in the matrix.\n\n");

}

return 0;

}



