**Assignment 06 2D-Array**

#include <iostream>

using *namespace* std;

*void* store(*int* \*\**arr*, *int* *rows*, *int* *cols*) {

    cout << "Enter elements in the matrix: \n";

    for (*int* i = 0; i < *rows*; i++) {

        for (*int* j = 0; j < *cols*; j++) {

            cin >> *arr*[i][j];

        }

    }

}

*void* display(*int* \*\**arr*, *int* *rows*, *int* *cols*) {

    cout << "Matrix elements:\n";

    for (*int* i = 0; i < *rows*; i++) {

        for (*int* j = 0; j < *cols*; j++) {

            cout << *arr*[i][j] << "\t";

        }

        cout << "\n";

    }

}

*int*\*\* createMatrix(*int* *rows*, *int* *cols*) {

*int* \*\*matrix = new *int*\*[*rows*];

    for (*int* i = 0; i < *rows*; i++) {

        matrix[i] = new *int*[*cols*]();

    }

    return matrix;

}

*void* freeMatrix(*int* \*\**matrix*, *int* *rows*) {

    for (*int* i = 0; i < *rows*; i++) {

        delete[] *matrix*[i];

    }

    delete[] *matrix*;

}

*void* addMatrices(*int* \*\**arr*, *int* \*\**brr*, *int* *rows*, *int* *cols*) {

*int* \*\*result = createMatrix(*rows*, *cols*);

    for (*int* i = 0; i < *rows*; i++) {

        for (*int* j = 0; j < *cols*; j++) {

            result[i][j] = *arr*[i][j] + *brr*[i][j];

        }

    }

    cout << "Addition Result:\n";

    display(result, *rows*, *cols*);

    freeMatrix(result, *rows*);

}

*void* subtractMatrices(*int* \*\**arr*, *int* \*\**brr*, *int* *rows*, *int* *cols*) {

*int* \*\*result = createMatrix(*rows*, *cols*);

    for (*int* i = 0; i < *rows*; i++) {

        for (*int* j = 0; j < *cols*; j++) {

            result[i][j] = *arr*[i][j] - *brr*[i][j];

        }

    }

    cout << "Subtraction Result:\n";

    display(result, *rows*, *cols*);

    freeMatrix(result, *rows*);

}

*void* transposeMatrix(*int* \*\**arr*, *int* *rows*, *int* *cols*) {

*int* \*\*transpose = createMatrix(*cols*, *rows*);

    for (*int* i = 0; i < *rows*; i++) {

        for (*int* j = 0; j < *cols*; j++) {

            transpose[j][i] = *arr*[i][j];

        }

    }

    cout << "Transpose of the Matrix:\n";

    display(transpose, *cols*, *rows*);

    freeMatrix(transpose, *cols*);

}

*void* multiplyMatrices(*int* \*\**A*, *int* \*\**B*, *int* *r1*, *int* *c1*, *int* *r2*, *int* *c2*) {

    if (*c1* != *r2*) {

        cout << "Matrix multiplication not possible.\n";

        return;

    }

*int* \*\*result = createMatrix(*r1*, *c2*);

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

        for (*int* j = 0; j < *c2*; j++) {

            for (*int* k = 0; k < *c1*; k++) {

                result[i][j] += *A*[i][k] \* *B*[k][j];

            }

        }

    }

    cout << "Multiplication Result:\n";

    display(result, *r1*, *c2*);

    freeMatrix(result, *r1*);

}

*int* main() {

*int* choice;

    do {

        cout << "\n--- 2D Array Operations ---\n";

        cout << "1. Addition\n";

        cout << "2. Subtraction\n";

        cout << "3. Transpose\n";

        cout << "4. Multiplication\n";

        cout << "0. Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        if (choice >= 1 && choice <= 4) {

*int* rows, cols, r2, c2;

*int* \*\*A, \*\*B;

            switch (choice) {

                case 1:

                case 2:

                    cout << "Enter rows and columns for both matrices: ";

                    cin >> rows >> cols;

                    A = createMatrix(rows, cols);

                    B = createMatrix(rows, cols);

                    store(A, rows, cols);

                    store(B, rows, cols);

                    display(A, rows, cols);

                    display(B, rows, cols);

                    if (choice == 1) addMatrices(A, B, rows, cols);

                    else subtractMatrices(A, B, rows, cols);

                    freeMatrix(A, rows);

                    freeMatrix(B, rows);

                    break;

                case 3:

                    cout << "Enter rows and columns of the matrix: ";

                    cin >> rows >> cols;

                    A = createMatrix(rows, cols);

                    store(A, rows, cols);

                    display(A, rows, cols);

                    transposeMatrix(A, rows, cols);

                    freeMatrix(A, rows);

                    break;

                case 4:

                    cout << "Enter rows and columns for Matrix A: ";

                    cin >> rows >> cols;

                    cout << "Enter rows and columns for Matrix B: ";

                    cin >> r2 >> c2;

                    A = createMatrix(rows, cols);

                    B = createMatrix(r2, c2);

                    store(A, rows, cols);

                    store(B, r2, c2);

                    display(A, rows, cols);

                    display(B, r2, c2);

                    multiplyMatrices(A, B, rows, cols, r2, c2);

                    freeMatrix(A, rows);

                    freeMatrix(B, r2);

                    break;

            }

        } else if (choice != 0) {

            cout << "Invalid choice! Please try again.\n";

        }

    } while (choice != 0);

    cout << "Exiting program. Goodbye!\n";

    return 0;

}

Output:

PS D:\Fullstack-Java-FirstBit-Solutions> & 'c:\Users\bhagv\.vscode\extensions\ms-vscode.cpptools-1.22.11-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-hbm2vt5j.lr2' '--stdout=Microsoft-MIEngine-Out-y43yu5ka.kmm' '--stderr=Microsoft-MIEngine-Error-msucuk4w.kaw' '--pid=Microsoft-MIEngine-Pid-dhdhchsn.14u' '--dbgExe=C:\TDM-GCC-64\bin\gdb.exe' '--interpreter=mi'

--- 2D Array Operations ---

1. Addition

2. Subtraction

3. Transpose

4. Multiplication

0. Exit

Enter your choice: 11

Invalid choice! Please try again.

--- 2D Array Operations ---

1. Addition

2. Subtraction

3. Transpose

4. Multiplication

0. Exit

Enter your choice:

1

Enter rows and columns for both matrices: 2

2

Enter elements in the matrix:

1

2 3 4

Enter elements in the matrix:

1 2 3 4

Matrix elements:

1 2

3 4

Matrix elements:

1 2

3 4

Addition Result:

Matrix elements:

2 4

6 8

--- 2D Array Operations ---

1. Addition

2. Subtraction

3. Transpose

4. Multiplication

0. Exit

Enter your choice: 2

Enter rows and columns for both matrices: 2 2

Enter elements in the matrix:

1 2 3 4

Enter elements in the matrix:

1 2 3 4

Matrix elements:

1 2

3 4

Matrix elements:

1 2

3 4

Subtraction Result:

Matrix elements:

0 0

0 0

--- 2D Array Operations ---

1. Addition

2. Subtraction

3. Transpose

4. Multiplication

0. Exit

Enter your choice: 3

Enter rows and columns of the matrix: 4 4

Enter elements in the matrix:

12 23 45 556 7 8 9 1 2 3 4 5 6 7 8 9

Matrix elements:

12 23 45 556

7 8 9 1

2 3 4 5

6 7 8 9

Transpose of the Matrix:

Matrix elements:

12 7 2 6

23 8 3 7

45 9 4 8

556 1 5 9

--- 2D Array Operations ---

1. Addition

2. Subtraction

3. Transpose

4. Multiplication

0. Exit

Enter your choice: 4

Enter rows and columns for Matrix A: 3 3

Enter rows and columns for Matrix B: 3 3

Enter elements in the matrix:

1 2 3 4 5 6 7 8 9

Enter elements in the matrix:

1 2 3 4 5 6 7 8 9

Matrix elements:

1 2 3

4 5 6

7 8 9

Matrix elements:

1 2 3

4 5 6

7 8 9

Multiplication Result:

Matrix elements:

30 36 42

66 81 96

102 126 150

--- 2D Array Operations ---

1. Addition

2. Subtraction

3. Transpose

4. Multiplication

0. Exit

Enter your choice: 0

Exiting program. Goodbye!

PS D:\Fullstack-Java-FirstBit-Solutions>