Cpp Practicals\Q8\Q8.cpp

```cpp
/*
8.  Create a Matrix class. Write a menu-driven program to perform following Matrix
operations (exceptions should be thrown by the functions if matrices passed to them
are incompatible and handled by the main() function):
a.  Sum
b.  Product
c.  Transpose

*/

#include <iostream>
#include <vector>
using namespace std;

class Matrix {
    int row, col;
    vector<vector<int>> arr;

public:
    Matrix() {}
    Matrix(int noOfRow, int noOfCol) : row(noOfRow), col(noOfCol), arr(noOfRow,
vector<int>(noOfCol, 0)) {}

    void inputMatrix() {
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                int element;
                cout << "Enter element at (" << i << "," << j << ") position: ";
                cin >> element;
                arr[i][j] = element;
            }
        }
    }

    void displayMatrix() const {
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                cout << arr[i][j] << " ";
            }
            cout << endl;
        }
    }

    int getElement(int i, int j) const {
        return arr[i][j];
    }

    void setElement(int i, int j, int ele) {
        arr[i][j] = ele;
    }

    friend Matrix sum(const Matrix&, const Matrix&);
    friend Matrix product(const Matrix&, const Matrix&);
    friend Matrix transpose(const Matrix&);
};

```

```
Matrix sum(const Matrix& A, const Matrix& B) {
    int row = A.row;
    int col = A.col;
    Matrix C(row, col);
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            C.setElement(i, j, A.getElement(i, j) + B.getElement(i, j));
        }
    }
    return C;
}

Matrix product(const Matrix& A, const Matrix& B) {
    int rowA = A.row;
    int colA = A.col;
    int rowB = B.row;
    int colB = B.col;

    if (colA != rowB) {
        throw "Matrix dimensions are not compatible for multiplication!";
    }

    Matrix C(rowA, colB);

    for (int i = 0; i < rowA; i++) {
        for (int j = 0; j < colB; j++) {
            int sum = 0;
            for (int k = 0; k < colA; k++) {
                sum += A.getElement(i, k) * B.getElement(k, j);
            }
            C.setElement(i, j, sum);
        }
    }

    return C;
}

Matrix transpose(const Matrix& A) {
    int row = A.row;
    int col = A.col;
    Matrix C(col, row);
    for (int i = 0; i < col; i++) {
        for (int j = 0; j < row; j++) {
            C.setElement(i, j, A.getElement(j, i));
        }
    }
    return C;
}

int main() {
    int rows, cols;

    cout << "Enter the number of rows in the matrices: ";
    cin >> rows;
    cout << "Enter the number of columns in the matrices: ";
    cin >> cols;

    Matrix A(rows, cols);
    Matrix B(rows, cols);

```

```
    cout << "Enter the elements of the first matrix:" << endl;
    A.inputMatrix();
    cout << "Enter the elements of the second matrix:" << endl;
    B.inputMatrix();

    int choice;
    cout << "Select an operation:" << endl;
    cout << "1. Sum" << endl;
    cout << "2. Product" << endl;
    cout << "3. Transpose" << endl;
    cout << "Enter your choice (1-3): ";
    cin >> choice;

    Matrix result;
    try {
        switch (choice) {
            case 1:
                result = sum(A, B);
                cout << "Sum of the matrices:" << endl;
                result.displayMatrix();
                break;
            case 2:
                result = product(A, B);
                cout << "Product of the matrices:" << endl;
                result.displayMatrix();
                break;
            case 3:
                result = transpose(A);
                cout << "Transpose of the matrix:" << endl;
                result.displayMatrix();
                break;
            default:
                cout << "Invalid choice!" << endl;
        }
    } catch (const char* errorMessage) {
        cout << "Error: " << errorMessage << endl;
    }

    return 0;
}


/*
Output:

PS C:\Users\hp\Desktop\Cpp\Cpp Practicals\Q8> cd "c:\Users\hp\Desktop\Cpp\Cpp
Practicals\Q8\" ; if ($?) { g++ Q8.cpp -o Q8 } ; if ($?) { .\Q8 }
Enter the number of rows in the matrices: 2
Enter the number of columns in the matrices: 2
Enter the elements of the first matrix:
Enter element at (0,0) position: 1
Enter element at (0,1) position: 1
Enter element at (1,0) position: 1
Enter element at (1,1) position: 1
Enter the elements of the second matrix:
Enter element at (0,0) position: 2
Enter element at (0,1) position: 2
Enter element at (1,0) position: 2
Enter element at (1,1) position: 2
Select an operation:
```

```
1. Sum
2. Product
3. Transpose
Enter your choice (1-3): 1
Sum of the matrices:
3 3
3 3

Select an operation:
1. Sum
2. Product
3. Transpose
Enter your choice (1-3): 2
Product of the matrices:
4 4
4 4

Select an operation:
1. Sum
2. Product
3. Transpose
Enter your choice (1-3): 3
Transpose of the matrix:
1 1
1 1
*/

```