

Cpp Practicals\Q8\Q8.cpp

```
1  /*
2  8. Create a Matrix class. Write a menu-driven program to perform following Matrix
3  operations (exceptions should be thrown by the functions if matrices passed to them
4  are incompatible and handled by the main() function):
5  a. Sum
6  b. Product
7  c. Transpose
8
9  */
10
11 #include <iostream>
12 #include <vector>
13 using namespace std;
14
15 class Matrix {
16     int row, col;
17     vector<vector<int>> arr;
18
19 public:
20     Matrix() {}
21     Matrix(int noOfRow, int noOfCol) : row(noOfRow), col(noOfCol), arr(noOfRow,
vector<int>(noOfCol, 0)) {}
22
23     void inputMatrix() {
24         for (int i = 0; i < row; i++) {
25             for (int j = 0; j < col; j++) {
26                 int element;
27                 cout << "Enter element at (" << i << ", " << j << ") position: ";
28                 cin >> element;
29                 arr[i][j] = element;
30             }
31         }
32     }
33
34     void displayMatrix() const {
35         for (int i = 0; i < row; i++) {
36             for (int j = 0; j < col; j++) {
37                 cout << arr[i][j] << " ";
38             }
39             cout << endl;
40         }
41     }
42
43     int getElement(int i, int j) const {
44         return arr[i][j];
45     }
46
47     void setElement(int i, int j, int ele) {
48         arr[i][j] = ele;
49     }
50
51     friend Matrix sum(const Matrix&, const Matrix&);
52     friend Matrix product(const Matrix&, const Matrix&);
53     friend Matrix transpose(const Matrix&);
54 };
55
```

```

56 Matrix sum(const Matrix& A, const Matrix& B) {
57     int row = A.row;
58     int col = A.col;
59     Matrix C(row, col);
60     for (int i = 0; i < row; i++) {
61         for (int j = 0; j < col; j++) {
62             C.setElement(i, j, A.getElement(i, j) + B.getElement(i, j));
63         }
64     }
65     return C;
66 }
67
68 Matrix product(const Matrix& A, const Matrix& B) {
69     int rowA = A.row;
70     int colA = A.col;
71     int rowB = B.row;
72     int colB = B.col;
73
74     if (colA != rowB) {
75         throw "Matrix dimensions are not compatible for multiplication!";
76     }
77
78     Matrix C(rowA, colB);
79
80     for (int i = 0; i < rowA; i++) {
81         for (int j = 0; j < colB; j++) {
82             int sum = 0;
83             for (int k = 0; k < colA; k++) {
84                 sum += A.getElement(i, k) * B.getElement(k, j);
85             }
86             C.setElement(i, j, sum);
87         }
88     }
89
90     return C;
91 }
92
93 Matrix transpose(const Matrix& A) {
94     int row = A.row;
95     int col = A.col;
96     Matrix C(col, row);
97     for (int i = 0; i < col; i++) {
98         for (int j = 0; j < row; j++) {
99             C.setElement(i, j, A.getElement(j, i));
100         }
101     }
102     return C;
103 }
104
105 int main() {
106     int rows, cols;
107
108     cout << "Enter the number of rows in the matrices: ";
109     cin >> rows;
110     cout << "Enter the number of columns in the matrices: ";
111     cin >> cols;
112
113     Matrix A(rows, cols);
114     Matrix B(rows, cols);
115

```

```

116     cout << "Enter the elements of the first matrix:" << endl;
117     A.inputMatrix();
118     cout << "Enter the elements of the second matrix:" << endl;
119     B.inputMatrix();
120
121     int choice;
122     cout << "Select an operation:" << endl;
123     cout << "1. Sum" << endl;
124     cout << "2. Product" << endl;
125     cout << "3. Transpose" << endl;
126     cout << "Enter your choice (1-3): ";
127     cin >> choice;
128
129     Matrix result;
130     try {
131         switch (choice) {
132             case 1:
133                 result = sum(A, B);
134                 cout << "Sum of the matrices:" << endl;
135                 result.displayMatrix();
136                 break;
137             case 2:
138                 result = product(A, B);
139                 cout << "Product of the matrices:" << endl;
140                 result.displayMatrix();
141                 break;
142             case 3:
143                 result = transpose(A);
144                 cout << "Transpose of the matrix:" << endl;
145                 result.displayMatrix();
146                 break;
147             default:
148                 cout << "Invalid choice!" << endl;
149         }
150     } catch (const char* errorMessage) {
151         cout << "Error: " << errorMessage << endl;
152     }
153
154     return 0;
155 }
156
157
158 /*
159 Output:
160
161 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 }
162 Enter the number of rows in the matrices: 2
163 Enter the number of columns in the matrices: 2
164 Enter the elements of the first matrix:
165 Enter element at (0,0) position: 1
166 Enter element at (0,1) position: 1
167 Enter element at (1,0) position: 1
168 Enter element at (1,1) position: 1
169 Enter the elements of the second matrix:
170 Enter element at (0,0) position: 2
171 Enter element at (0,1) position: 2
172 Enter element at (1,0) position: 2
173 Enter element at (1,1) position: 2
174 Select an operation:

```

```
175 | 1. Sum
176 | 2. Product
177 | 3. Transpose
178 | Enter your choice (1-3): 1
179 | Sum of the matrices:
180 | 3 3
181 | 3 3
182 |
183 | Select an operation:
184 | 1. Sum
185 | 2. Product
186 | 3. Transpose
187 | Enter your choice (1-3): 2
188 | Product of the matrices:
189 | 4 4
190 | 4 4
191 |
192 | Select an operation:
193 | 1. Sum
194 | 2. Product
195 | 3. Transpose
196 | Enter your choice (1-3): 3
197 | Transpose of the matrix:
198 | 1 1
199 | 1 1
200 | */
201 |
```