

MÀN HÌNH CONSOLE:

```
E:\hoc\OOP\Code\lapchong+ X + v
Nhap ma tran thu nhat:
Nhap so hang: 2
Nhap so cot: 2
Nhap cac phan tu cua ma tran:
Nhap phan tu thu [0][0]: 1
Nhap phan tu thu [0][1]: 2
Nhap phan tu thu [1][0]: 3
Nhap phan tu thu [1][1]: 4
Nhap ma tran thu hai:
Nhap so hang: 2
Nhap so cot: 2
Nhap cac phan tu cua ma tran:
Nhap phan tu thu [0][0]: 2
Nhap phan tu thu [0][1]: 4
Nhap phan tu thu [1][0]: 6
Nhap phan tu thu [1][1]: 8
Ma tran thu nhat:
Ma tran:
1 2
3 4
Ma tran thu hai:
Ma tran:
2 4
6 8
[INFO]: phiep cong ma tran
Ket qua phiep cong ma tran:
Ma tran:
3 6
9 12
[INFO]: Phep tru ma tran
Ket qua phep tru ma tran:
Ma tran:
-1 -2
-3 -4
[INFO]: Nhan ma tran
Ket qua phep nhan ma tran:
Ma tran:
14 20
30 44
[ERROR]: Khong co phep chia
```

```
Ma tran thu nhat:
Ma tran:
1 2
3 4
Ma tran thu hai:
Ma tran:
2 4
6 8
[INFO]: phep cong ma tran
Ket qua phep cong ma tran:
Ma tran:
3 6
9 12
[INFO]: Phep tru ma tran
Ket qua phep tru ma tran:
Ma tran:
-1 -2
-3 -4
[INFO]: Nhan ma tran
Ket qua phep nhan ma tran:
Ma tran:
14 20
30 44
[ERROR]: Khong co phep chia
[DEBUG]: Ban co the su dung ma tran nghich dao
Ket qua phep chia ma tran:

-----
Process exited after 9.761 seconds with return value 0
Press any key to continue . . . |
```

CODE:

```
matrixlogging.cpp
1  #include <iostream>
2  #include <vector>
3  #include<string>
4
5  using namespace std;
6
7  class Logger {
8  public:
9      static void log(const string& message, int check) {
10         if(check==0){
11             cout <<"[INFO]: " << message << endl;
12         }else if(check==1){
13             cout <<"[DEBUG]: " << message << endl;
14         }else if(check==2){
15             cout <<"[ERROR]: " << message << endl;
16         }
17     }
18 };
19
20 class Matrix {
21 private:
22     int m;
23     int n;
24     double elements[100][100];
25
26 public:
27     Matrix() : m(0), n(0) {}
28
29     Matrix(const Matrix& a) : m(a.m), n(a.n) {
30         for (int i = 0; i < m; i++) {
31             for (int j = 0; j < n; j++) {
32                 elements[i][j] = a.elements[i][j];
33             }
34         }
35     }
36 }
```

```
22     int m;  
23     int n;  
24     double elements[100][100];  
25  
26 public:  
27     Matrix() : m(0), n(0) {}  
28  
29     Matrix(const Matrix& a) : m(a.m), n(a.n) {  
30         for (int i = 0; i < m; i++) {  
31             for (int j = 0; j < n; j++) {  
32                 elements[i][j] = a.elements[i][j];  
33             }  
34         }  
35     }  
36  
37     void nhap() {  
38         cout << "Nhap so hang: ";  
39         cin >> m;  
40         cout << "Nhap so cot: ";  
41         cin >> n;  
42  
43         cout << "Nhap cac phan tu cua ma tran:" << endl;  
44         for (int i = 0; i < m; i++) {  
45             for (int j = 0; j < n; j++) {  
46                 cout << "Nhap phan tu thu [" << i << "][" << j << "]: ";  
47                 cin >> elements[i][j];  
48             }  
49         }  
50     }  
51  
52     void xuat() const {  
53         cout << "Ma tran:" << endl;  
54         for (int i = 0; i < m; i++) {  
55             for (int j = 0; j < n; j++) {  
56                 cout << elements[i][j] << " ";  
57             }
```

```
49     }
50 }
51
52 void xuat() const {
53     cout << "Ma tran:" << endl;
54     for (int i = 0; i < m; i++) {
55         for (int j = 0; j < n; j++) {
56             cout << elements[i][j] << " ";
57         }
58         cout << endl;
59     }
60 }
61
62 Matrix operator+(const Matrix& a) const {
63     Matrix result(*this);
64     if (m == a.m && n == a.n) {
65         for (int i = 0; i < m; i++) {
66             for (int j = 0; j < n; j++) {
67                 result.elements[i][j] += a.elements[i][j];
68             }
69         }
70         Logger::log("phep cong ma tran",0);
71         return result;
72     }
73     Logger::log("Khong the cong 2 ma tran khong cung kich thuc",2);
74     Logger::log("Vui long nhap lai ma tran",1);
75     return *this;
76 }
77
78 Matrix operator-(const Matrix& a) const {
79     Matrix result(*this);
80     if (m == a.m && n == a.n) {
81         for (int i = 0; i < m; i++) {
82             for (int j = 0; j < n; j++) {
83                 result.elements[i][j] -= a.elements[i][j];
84             }
85         }
86     }
87     return result;
88 }
```

```

79     Matrix result(*this);
80     if (m == a.m && n == a.n) {
81         for (int i = 0; i < m; i++) {
82             for (int j = 0; j < n; j++) {
83                 result.elements[i][j] -= a.elements[i][j];
84             }
85         }
86         Logger::log("Phep tru ma tran",0);
87         return result;
88     }
89     Logger::log("Khong the tru 2 ma tran khong cung kich thuoc",2);
90     Logger::log("Vui long nhap lai ma tran",1);
91     return *this;
92 }
93
94 Matrix operator*(const Matrix& a) const {
95     Matrix result;
96     if (n == a.m) {
97         result.m = m;
98         result.n = a.n;
99         for (int i = 0; i < m; i++) {
100             for (int j = 0; j < a.n; j++) {
101                 result.elements[i][j] = 0;
102                 for (int k = 0; k < n; k++) {
103                     result.elements[i][j] += elements[i][k] * a.elements[k][j];
104                 }
105             }
106         }
107         Logger::log("Nhan ma tran",0);
108         return result;
109     }
110     Logger::log("Khong nhan duoc 2 ma tran khac kich thuoc",2);
111     Logger::log("Vui long nhap lai ma tran",1);
112     return result;
113 }

```

matrixlogging.cpp

```
109     }
110     Logger::log("Khong nhan duoc 2 ma tran khac kich thuoc",2);
111     Logger::log("Vui long nhap lai ma tran",1);
112     return result;
113 }
114
115 Matrix operator/(const Matrix& a) const {
116     Logger::log("Khong co phep chia",2);
117     Logger::log("Ban co the su dung ma tran nghich dao",1);
118     return *this;
119 }
120 };
121
122 int main() {
123     Matrix mt1, mt2;
124     cout << "Nhap ma tran thu nhat:" << endl;
125     mt1.nhap();
126     cout << "Nhap ma tran thu hai:" << endl;
127     mt2.nhap();
128
129     cout << "Ma tran thu nhat:" << endl;
130     mt1.xuat();
131     cout << "Ma tran thu hai:" << endl;
132     mt2.xuat();
133
134     Matrix result;
135
136     result = mt1 + mt2;
137     cout << "Ket qua phep cong ma tran:" << endl;
138     result.xuat();
139
140     result = mt1 - mt2;
141     cout << "Ket qua phep tru ma tran:" << endl;
142     result.xuat();
143
144     result = mt1 * mt2;
```

matrixlogging.cpp

```
117 |         Logger::log("Ban co the su dung ma tran nghich dao",1);
118 |         return *this;
119 |     }
120 | };
121 |
122 | int main() {
123 |     Matrix mt1, mt2;
124 |     cout << "Nhap ma tran thu nhat:" << endl;
125 |     mt1.nhap();
126 |     cout << "Nhap ma tran thu hai:" << endl;
127 |     mt2.nhap();
128 |
129 |     cout << "Ma tran thu nhat:" << endl;
130 |     mt1.xuat();
131 |     cout << "Ma tran thu hai:" << endl;
132 |     mt2.xuat();
133 |
134 |     Matrix result;
135 |
136 |     result = mt1 + mt2;
137 |     cout << "Ket qua phep cong ma tran:" << endl;
138 |     result.xuat();
139 |
140 |     result = mt1 - mt2;
141 |     cout << "Ket qua phep tru ma tran:" << endl;
142 |     result.xuat();
143 |
144 |     result = mt1 * mt2;
145 |     cout << "Ket qua phep nhan ma tran:" << endl;
146 |     result.xuat();
147 |
148 |     result = mt1 / mt2;
149 |     cout << "Ket qua phep chia ma tran:" << endl;
150 |     return 0;
151 | }
```