

|  |
| --- |
| **EX11-02a.cpp:** *Multiply Matrices* |
| **Line#** | **Code** |
| 123456789101112131415161718192021222324252627282930313233343536373839404142434445464748495051 | #include <iostream>using namespace std;#define NOI(\_arr) (sizeof(\_arr)/sizeof(\_arr[0]))#define ROW(\_m) NOI(\_m)#define COL(\_m) NOI(\_m[0])#define showMatrix(\_m) \_showMatrix((double\*)&\_m, ROW(\_m), COL(\_m))void \_showMatrix(double\* m, int row, int col) { for (int c = 0; c < col; c++) cout << "------"; cout << endl; for (int r = 0; r < row; r++) { for (int c = 0; c < col; c++) { cout << m[r \* col + c] << '\t'; } cout << endl; } for (int c = 0; c < col; c++) cout << "------"; cout << endl;}int main() { double mA[][2] = { {1,2}, {3,4}, {5,6}, }; double mB[][2] = { {1,2}, {3,4}, }; if (COL(mA) != ROW(mB)) { cout << "Cannot Multiply-lah!" << endl; } else { double mC[ROW(mA)][COL(mB)]; for (int r = 0; r < ROW(mC); r++) { for (int c = 0; c < COL(mC); c++) { mC[r][c] = 0; for (int i = 0; i < COL(mA); i++) {//For Sum-Product mC[r][c] += mA[r][i] \* mB[i][c]; } } } showMatrix(mA); cout << 'X' << endl; showMatrix(mB); cout << '=' << endl; showMatrix(mC); } return 0;} |



|  |
| --- |
| **EX11-02b.cpp:** *Add Matrices* |
| **Line#** | **Code** |
| 123456789101112131415161718192021222324252627282930313233343536373839404142434445464748495051 | #include <iostream>using namespace std;#define NOI(\_arr) (sizeof(\_arr)/sizeof(\_arr[0]))#define ROW(\_m) NOI(\_m)#define COL(\_m) NOI(\_m[0])#define showMatrix(\_m) \_showMatrix((double\*)&\_m, ROW(\_m), COL(\_m))void \_showMatrix(double\* m, int row, int col) { for (int c = 0; c < col; c++) cout << "------"; cout << endl; for (int r = 0; r < row; r++) { for (int c = 0; c < col; c++) { cout << m[r \* col + c] << '\t'; } cout << endl; } for (int c = 0; c < col; c++) cout << "------"; cout << endl;}int main() { double mA[][2] = { {1,2}, {3,4}, {5,6}, }; double mB[][2] = { {1,1}, {2,2}, {3,3}, }; if ((ROW(mA) != ROW(mB)) || (COL(mA) != COL(mB))) { cout << "Cannot Add-lah!" << endl; } else { double mC[ROW(mA)][COL(mA)]; for (int r = 0; r < ROW(mC); r++) { for (int c = 0; c < COL(mC); c++) { mC[r][c] = mA[r][c] + mB[r][c]; } } showMatrix(mA); cout << '+' << endl; showMatrix(mB); cout << '=' << endl; showMatrix(mC); } return 0;} |



|  |
| --- |
| **EX11-02c.cpp:** *Transpose Matrix* |
| **Line#** | **Code** |
| 12345678910111213141516171819202122232425262728293031323334353637383940414243 | #include <iostream>#include <io.h>#include <fcntl.h>using namespace std;#define NOI(\_arr) (sizeof(\_arr)/sizeof(\_arr[0]))#define ROW(\_m) NOI(\_m)#define COL(\_m) NOI(\_m[0])#define showMatrix(\_m) \_showMatrix((double\*)&\_m, ROW(\_m), COL(\_m))void \_showMatrix(double\* m, int row, int col) { for (int c = 0; c < col; c++) cout << "------"; cout << endl; for (int r = 0; r < row; r++) { for (int c = 0; c < col; c++) { cout << m[r \* col + c] << '\t'; } cout << endl; } for (int c = 0; c < col; c++) cout << "------"; cout << endl;}int main() { double mA[][2] = { {1,2}, {3,4}, {5,6}, }; double mAt[COL(mA)][ROW(mA)]; for (int r = 0; r < ROW(mAt); r++) { for (int c = 0; c < COL(mAt); c++) { mAt[r][c] = mA[c][r]; } } showMatrix(mA); \_setmode(\_fileno(stdout), \_O\_U16TEXT); wcout << L"\u2193" << endl; \_setmode(\_fileno(stdout), \_O\_TEXT); showMatrix(mAt); return 0;} |