

Mátrixok használata

1. feladat: Adjunk össze két 3×3 -as egész számokat tároló mátrix-ot és tároljunk le egy harmadikban az eredményt. Használjunk függvényt a mátrix elemeinek bekéréséhez.

```
#include <stdio.h>

#define SIZE 3 // a matrix-ok merete
void readMatrix(int m[][SIZE])
{
    for(int sor = 0; sor < SIZE; sor++)
    {
        for(int oszlop = 0; oszlop < SIZE; oszlop++)
        {
            scanf("%d", &m[sor][oszlop]);
        }
    }
}

int main()
{
    int A[SIZE][SIZE]; // Matrix 1
    int B[SIZE][SIZE]; // Matrix 2
    int C[SIZE][SIZE]; // eredmeny Matrix

    printf("Kerem az elso matrix elemeit: \n");
    readMatrix(A);
    printf("Kerem az masodik matrix elemeit: \n");
    readMatrix(B);

    for(int sor = 0; sor < SIZE; sor++)
    {
        for(int oszlop = 0; oszlop < SIZE; oszlop++)
        {
            C[sor][oszlop] = A[sor][oszlop] + B[sor][oszlop];
        }
    }

    printf("\nMatrixok osszege: A+B = \n");
    for(int sor = 0; sor < SIZE; sor++)
    {
        for(int oszlop = 0; oszlop < SIZE; oszlop++)
        {
            printf("%d ", C[sor][oszlop]);
        }
        printf("\n"); // soronkent uj sor
    }
}
```

2. feladat: Az első feladatban a

```
readMatrix()
```

függvényt cseréljük le:

```
void readMatrix(int ** m)
```

-ra és próbáljuk meg mi történik és magyarázzuk meg miért?

3. feladat: Az első feladatban megadott feladatot oldjuk meg pointerekkel is.

```
#include <stdio.h>

#define SIZE 3 // a matrix-ok merete
void readMatrix(int *m)
{
    for(int i = 0; i < SIZE * SIZE; i++)
    {
        scanf("%d", (m + i));
    }
}

int main()
{
    int A[SIZE][SIZE]; // Matrix 1
    int B[SIZE][SIZE]; // Matrix 2

    printf("Kerem az elso matrix elemeit: \n");
    readMatrix(&A[0][0]);
    printf("Kerem az masodik matrix elemeit: \n");
    readMatrix(&B[0][0]);

    int C[SIZE][SIZE]; // eredmeny Matrix

    for(int i = 0; i < SIZE * SIZE; i++)
    {
        *(&C[0][0] + i) = *(&A[0][0] + i) + *(&B[0][0] + i);
    }

    printf("\nMatrixok osszege: A+B = \n");
    for(int sor = 0; sor < SIZE; sor++)
    {
        for(int oszlop = 0; oszlop < SIZE; oszlop++)
        {
            printf("%d ", C[sor][oszlop]);
        }
    }
}
```

```
        printf("\n"); // soronként új sor
    }
}
```

4. feladat: Ellenőrizzük, hogy a felhasználótól bekért 3×3-as mátrix felső háromszög típusú-e?

```
#include <stdio.h>

#define SIZE 3 // a matrix-ok merete
void readMatrix(int m[][SIZE])
{
    for(int sor = 0; sor < SIZE; sor++)
    {
        for(int oszlop = 0; oszlop < SIZE; oszlop++)
        {
            scanf("%d", &m[sor][oszlop]);
        }
    }
}

int main()
{
    int A[SIZE][SIZE];

    printf("Kerem a matrix elemeit: \n");
    readMatrix(A);

    int felsoHaromszog = 1;
    // Hogyan jovunk ra a megoldasra?
    //
    // 1 2 3   [0,0] [0,1] [0,2]
    // 0 4 5 => [1,0] [1,1] [1,2]
    // 0 0 6   [2,0] [2,1] [2,2]
    //
    // Mi a közös a három 0 érték indexeiben?
    // - az hogy a sor index erteke mindig nagyobb mint
    // az oszlop index!!
    //
    for(int sor = 0; sor < SIZE; sor++)
    {
        for(int oszlop = 0; oszlop < SIZE; oszlop++)
        {
            //
            // a foatlo alatti elemeknek 0-nak kell lenniuk
            //
            if(sor > oszlop && A[sor][oszlop] != 0)
            {
                felsoHaromszog = 0;
            }
        }
    }
}
```

```
    }  
}  
if(felsőHaromszog)  
{  
    printf("felso haromszog matrix");  
}  
else  
{  
    printf("nem felso haromszog");  
}  
}
```

5. feladat: Írjon C programot, ami meghatározza, hogy egy mátrix ritka mátrix-e? Egy ritka mátrix olyan mátrix aminek legalább elemeinek a fele 0.

6. feladat: Írjon C programot, ami bekér a felhasználótól egy 3×3-as mátrixot és kiírja a transzponáltját.

```
#include <stdio.h>  
  
#define SIZE 3 // a matrix-ok merete  
void readMatrix(int m[][SIZE])  
{  
    for(int sor = 0; sor < SIZE; sor++)  
    {  
        for(int oszlop = 0; oszlop < SIZE; oszlop++)  
        {  
            scanf("%d", &m[sor][oszlop]);  
        }  
    }  
}  
  
int main()  
{  
    int A[SIZE][SIZE];  
    int E[SIZE][SIZE];  
    printf("Kerem a matrix elemeit: \n");  
    readMatrix(A);  
  
    // Hogyan jovunk ra a megoldasra?  
    // Transzpontalt matrix pelda:  
    //  
    // 1 2 3    1 4 7  
    // 4 5 6 => 2 5 8  
    // 7 8 9    3 6 9  
    //
```

```
// A matrix [i][j] elemét [j][i]-re kell cserélni
for(int sor = 0; sor < SIZE; sor++)
{
    for(int oszlop = 0; oszlop < SIZE; oszlop++)
    {
        E[sor][oszlop] = A[oszlop][sor];
    }
}

for(int sor = 0; sor < SIZE; sor++)
{
    for(int oszlop = 0; oszlop < SIZE; oszlop++)
    {
        printf("%d ", E[sor][oszlop]);
    }
    printf("\n");
}
}
```

7. feladat: Írjon C programot, ami összeszoroz két mátrixot.

```
#include <stdio.h>

#define MAX_SOR 3
#define MAX_OSZLOP 3
void readMatrix(int m[][MAX_OSZLOP])
{
    for(int sor = 0; sor < MAX_SOR; sor++)
    {
        for(int oszlop = 0; oszlop < MAX_OSZLOP; oszlop++)
        {
            scanf("%d", &m[sor][oszlop]);
        }
    }
}

int main()
{
    int A[MAX_SOR][MAX_OSZLOP];
    int B[MAX_SOR][MAX_OSZLOP];
    int C[MAX_SOR][MAX_OSZLOP];
    printf("Kerem az A matrix elemeit: \n");
    readMatrix(A);
    printf("Kerem a B matrix elemeit: \n");
    readMatrix(B);

    for (int sor = 0; sor < MAX_SOR; sor++)
    {
```

```
        for (int oszlop = 0; oszlop < MAX_OSZLOP; oszlop++)
        {
            int sum = 0;
            for (int i = 0; i < MAX_OSZLOP; i++)
            {
                sum += (*(A + sor) + i) * (*(B + i) + oszlop));
            }
            *(C + sor) + oszlop) = sum;
        }
    }

    for(int sor = 0; sor < MAX_SOR; sor++)
    {
        for(int oszlop = 0; oszlop < MAX_OSZLOP; oszlop++)
        {
            printf("%d ", C[sor][oszlop]);
        }
        printf("\n");
    }
}
```

8. feladat: Írja át az előző példában a `sum += (*(A + sor .... -t az egyszerűbb indexes A[i][j]- alakra!`

```
sum += A[sor][i] * B[i][oszlop];
```

9. feladat: Oldjuk meg nem négyzetes mátrixokkal is a feladatot, valamint dinamikus memória kezeléssel:

```
#include <stdio.h>
#include <stdlib.h>

// Dinamikus mátrix létrehozás
int **allocateMatrix(int rows, int cols) {
    int **matrix = (int **)malloc(rows * sizeof(int *));
    for (int i = 0; i < rows; i++) {
        matrix[i] = (int *)malloc(cols * sizeof(int));
    }
    return matrix;
}

// Mátrix felszabadítás
void freeMatrix(int **matrix, int rows) {
    for (int i = 0; i < rows; i++) {
        free(matrix[i]);
    }
}
```

```
    }
    free(matrix);
}

// Mátrix beolvasás
void readMatrix(int **m, int rows, int cols) {
    printf("Adja meg a mátrix elemeit (%d x %d):\n", rows, cols);
    for (int sor = 0; sor < rows; sor++) {
        for (int oszlop = 0; oszlop < cols; oszlop++) {
            scanf("%d", &m[sor][oszlop]);
        }
    }
}

// Mátrixok összeadása
void addMatrices(int **A, int **B, int **C, int rows, int cols) {
    for (int sor = 0; sor < rows; sor++) {
        for (int oszlop = 0; oszlop < cols; oszlop++) {
            C[sor][oszlop] = A[sor][oszlop] + B[sor][oszlop];
        }
    }
}

void printMatrix(int **matrix, int rows, int cols) {
    printf("\nMátrix:\n");
    for (int sor = 0; sor < rows; sor++) {
        for (int oszlop = 0; oszlop < cols; oszlop++) {
            printf("%d ", matrix[sor][oszlop]);
        }
        printf("\n");
    }
}

int main() {
    int rows, cols;

    printf("Adja meg a mátrix sorainak és oszlopainak számát: ");
    scanf("%d %d", &rows, &cols);

    int **A, **B, **C;

    // Mátrixok allokalása a megadott méret szerint
    A = allocateMatrix(rows, cols);
    B = allocateMatrix(rows, cols);
    C = allocateMatrix(rows, cols);

    // Mátrixok beolvasása
    printf("Kerem az elso matrix elemeit:\n");
    readMatrix(A, rows, cols);

    printf("Kerem a masodik matrix elemeit:\n");
```

```
readMatrix(B, rows, cols);

// Összeadás
addMatrices(A, B, C, rows, cols);

printf("\nMatrixok osszege: A+B = \n");
printMatrix(C, rows, cols);

// Memória felszabadítása
freeMatrix(A, rows);
freeMatrix(B, rows);
freeMatrix(C, rows);

return 0;
}
```

9. feladat : vizsgáljuk meg a llama chat model c implementációját:

<https://github.com/karpathy/llama2.c/blob/master/run.c> a 217.es sortól kezdődő függvényt.

From:

<https://edu.iit.uni-miskolc.hu/> - Institute of Information Science - University of Miskolc

Permanent link:

https://edu.iit.uni-miskolc.hu/tanszek:oktatas:muszaki_informatika:matrixok_kezelese?rev=1739354104

Last update: 2025/02/12 09:55

