Vektorok

Mi az a vektor (tömb)?

- A tömb egy azonos típusú adatok sorozata, amelyet egyetlen névvel hivatkozhatunk.
- Előnye: sok adatot tudunk tárolni és kezelni egy változó segítségével.
- A tömb elemei indexeléssel érhetők el (0-tól kezdődően!).

Létrehozás

```
int a[5];       // 5 elemű egész szám tömb (index: 0..4)
float b[10];    // 10 elemű valós szám tömb
```

Inicializálás

```
int x[5] = {1, 2, 3, 4, 5};     // pontosan 5 elem
int y[] = {10, 20, 30};         // a méretet a fordító számolja ki (3)
```

Elérés

```
int t[5] = {2, 4, 6, 8, 10};
printf("%d", t[2]); // a 3. elem, azaz 6
t[0] = 99;          // első elem átírása
```

Beolvasás / kiírás

```
int n = 5;
int v[5];
for (int i = 0; i < n; i++) {
    scanf("%d", &v[i]);      // beolvasás
}
for (int i = 0; i < n; i++) {
    printf("%d ", v[i]);     // kiírás
}
```

Tipikus műveletek vektorokon

- Összegzés: minden elem összeadása
- Maximum/minimum keresés: legnagyobb / legkisebb elem
- Keresés: adott érték előfordul-e
- Megfordítás: elemek sorrendjének felcserélése
- Másolás: egyik tömbből másikba

Mátrixok

A mátrix egy kétdimenziós vektor, ezek kívül jellemzői megegyeznek a vektorral. Mindig sor-oszlop sorrendben adjuk meg.

Létrehozás

```
int m[3][4];   // 3 sor, 4 oszlop (összesen 12 elem)
```

Inicializálás

```
int mat[2][3] = {
    {1, 2, 3},
    {4, 5, 6}
};
```

Elérés

```
printf("%d", mat[1][2]); // 2. sor, 3. oszlop → 6
mat[0][0] = 9;           // bal felső elem átírása
```

Beolvasás / kiírás

```
int sor = 3, oszlop = 3;
int m[3][3];

for (int i = 0; i < sor; i++) {
    for (int j = 0; j < oszlop; j++) {
        scanf("%d", &m[i][j]);
    }
}
for (int i = 0; i < sor; i++) {
    for (int j = 0; j < oszlop; j++) {
        printf("%d ", m[i][j]);
    }
    printf("\n");
}
```

Tipikus műveletek mátrixokon

- Sor / oszlop összege
- Főátló és mellékátló összege (négyzetes mátrixnál)
- Transzponálás (sorok ↔ oszlopok felcserélése)
- Mátrixösszeadás (azonos méretűek összeadása)
- Mátrixszorzás (haladóbb)

Gyakorlás

1. Írj programot, ami bekér n egész számot (n>0), majd egy int tömböt n elemmel, kiszámolja és kiírja az elemek összegét és átlagát (lebegőpontosan).

```
#include <stdio.h>

int main(){
    int n;
    printf("Mekkora legyen a vektor?");
    scanf("%d", &n);
    int v[n];
    int osszeg = 0;
    double atlag = 0;
    for(int i = 0; i < n; i++){
        printf("Adja meg a vektor %d. elemét:", i+1);
        scanf("%d", &v[i]);
        osszeg += v[i];
    }
    atlag = (double)osszeg / n;
    pritnf("\nA vektor elemeinek összege: %d, átlaga: %d", osszeg, atlag);
    return 0;
  }
```

2. Írj programot, ami bekér n számot és tömböt; írd ki a legkisebb és legnagyobb elem értékét és az első előfordulásuk indexét.

```
#include <stdio.h>

int main(void) {
    int n;
    printf("Hány eleme legyen a vektornak?");
    scanf("%d", &n);
    int v[n];
    for (int i = 0; i < n; ++i){
        printf("\nAdja meg a vektor %d. elemét: ", i+1);
        scanf("%d", &v[i]);
    }

    int min = v[0], max = v[0];
```

```
    int minIdx = 0, maxIdx = 0;
    for (int i = 1; i < n; ++i) {
        if (v[i] < min) {
            min = v[i];
            minIdx = i;
        }
        if (v[i] > max) {
            max = v[i];
            maxIdx = i;
        }
    }
    printf("Min: %d (%d. helyen)\nMax: %d (%d. helyen)\n", min, minIdx+1,
max, maxIdx+1);
    return 0;
}
```

3. Olvass be tömböt és egy keresett értéket x. Írd ki, hányszor fordul elő x a tömbben, és az összes indexét.

```
#include <stdio.h>

int main(void) {
    int n, x;
    int szamlalo = 0;
    printf("Hány eleme legyen a vektornak?");
    scanf("%d", &n);
    int v[n];
    for (int i = 0; i < n; ++i){
        printf("\nAdja meg a vektor %d. elemét: ", i+1);
        scanf("%d", &v[i]);
    }
    printf("Melyik számot keresi?");
    scanf("%d", &x);

    for (int i = 0; i < n; ++i)
        if (v[i] == x) {
            szamlalo++;
        }
    if (szamlalo == 0){ printf("Nincs ilyen szám benne.\n");}
    else {printf("\A %d szám %dx szerepel a vektorban.", x, szamlalo );}
    return 0;
}
```

4. Válaszd szét egy tömb elemeit páros és páratlan tömbökbe; írd ki a két új tömb méretét és elemeit.

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

```
int main(void) {
    int n;
    printf("Hány eleme legyen a vektornak?");
    scanf("%d", &n);

    int v[n];
    for (int i = 0; i < n; i++){
        scanf("%d", &v[i]);
    }

    int paros[n], paratlan[n];
    int parosSzamlalo = 0, paratlanSzamlalo = 0;

    for (int i = 0; i < n; i++) {
        if (v[i] % 2 == 0) {
            paros[parosSzamlalo++] = v[i];
        } else {
            paratlan[paratlanSzamlalo++] = v[i];
        }
    }

    printf("Páros (%d):", parosSzamlalo);
    for (int i = 0; i < parosSzamlalo; i++){
        printf(" %d", paros[i]);
    }

    printf("\nPáratlan (%d):", paratlanSzamlalo);
    for (int i = 0; i < paratlanSzamlalo; i++){
        printf(" %d", paratlan[i]);
    }

    return 0;
}
```

5. Két tömb (n elemű) összehasonlítása: mondja meg, megegyeznek-e, ha nem, az első különböző indexet.

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

int main(void) {
    int n;
    printf("Mekkorák legyenek a vektorok?");
    scanf("%d", &n);
    int a[n], b[n];
    for (int i = 0; i < n; ++i){
        printf("\nAdja meg az A vektor %d. elemét: ", i+1);
        scanf("%d", &a[i]);
    }
```

```
    for (int i = 0; i < n; ++i){
        printf("\nAdja meg a B vektor %d. elemét: ", i+1);
        scanf("%d", &b[i]);
    }

    for (int i = 0; i < n; ++i) {
        if (a[i] != b[i]) {
            printf("A %d. helyen különböznek: %d vs %d\n", i+1, a[i], b[i]);
            return 0;
        }
    }
    printf("Ugyanaz a kettő vektor.\n");
    return 0;
}
```

6. Implementálj buborék rendezést növekvő sorrendre, majd írd ki a rendezett tömböt.

```
#include <stdio.h>

int main(void) {
    int n;
    printf("Mekkorák legyen a vektor?");
    scanf("%d", &n);

    int v[n];
    for (int i = 0; i < n; i++){
        printf("\nAdja meg a vektor %d. elemét: ", i+1);
        scanf("%d", &v[i]);
    }

    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - 1 - i; j++) {
            if (v[j] > v[j + 1]) {
                int seged = v[j];
                v[j] = v[j + 1];
                v[j + 1] = seged;
            }
        }
    }

    printf("Rendezve: ");
    for (int i = 0; i < n; i++){
        printf("%d ", v[i]);
    }

    return 0;
}
```

7. Olvass be egy m x n egész mátrixot és írd ki a transzponáltját (n x m).

```
#include <stdio.h>

int main(void) {
    int m, n;
    printf("Mekkora legyen a mátrix? (n x m)");
    scanf("%d %d", &m, &n);

    int a[m][n];         // eredeti
    int t[n][m];         // transzponált

    // beolvasás
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &a[i][j]);
        }
    }

    // transzponálás
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            t[i][j] = a[j][i];
        }
    }

    // kiírás
    printf("Transzponálva:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            printf("%d ", t[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

8. Készíts programot, ami kiírja minden sor és minden oszlop összegét egy m x n mátrixon.

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

int main(void) {
    int m, n;
    printf("Mekkora legyen a mátrix? (m x n)");
    scanf("%d %d", &m, &n);
    int a[m][n];
    for (int i = 0; i < m; ++i){
```

```
        for (int j = 0; j < n; ++j){
            printf("%d sor %d oszlop:", i+1, j+1);
            scanf("%d", &a[i][j]);
        }
    }
    for (int i = 0; i < m; ++i) {
        int s = 0;
        for (int j = 0; j < n; ++j){
            s += a[i][j];
        }
        printf("A %d. sor összege = %d\n", i+1, s);
    }

    for (int j = 0; j < n; ++j) {
        int s = 0;
        for (int i = 0; i < m; ++i){
            s += a[i][j];
        }
        printf("A %d. oszlop összege = %d\n", j+1, s);
    }
    return 0;
}
```

9. Implementálj A(m×p) * B(p×n) = C(m×n) mátrixszorzást. Ellenőrizd a dimenziókat.

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

int main(void) {
    int m, p, p2, n;
    printf("Mekkora legyen az első mátrix? (m x n)");
    scanf("%d %d", &m, &p);
    int A[m][p];
    for (int i = 0; i < m; ++i){
        for (int j = 0; j < p; ++j){
            scanf("%d", &A[i][j]);
        }
    }
    printf("Mekkora legyen a második mátrix? (m x n)");
    scanf("%d %d", &p2, &n);
    if (p2 != p) {
        printf("Lehetetlen, különbözik a két mátrix mérete.\n");
        return 0;
    }
    int B[p][n];
    for (int i = 0; i < p; ++i){
        for (int j = 0; j < n; ++j){
            scanf("%d", &B[i][j]);
```

```
        }
    }

    int C[m][n]; //az eredmény tárolására
    for (int i = 0; i < m; ++i){
        for (int j = 0; j < n; ++j) {
            int sum = 0;
            for (int k = 0; k < p; ++k){
                sum += A[i][k] * B[k][j];
            }
            C[i][j] = sum;
        }
    }
    //Eredmény kiírása
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j){
            printf("%d ", C[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

10. Adott n x n mátrixra számold ki a főátló és mellékátló összegét, és írd ki.

```
#include <stdio.h>

int main(void) {
    int n;
    scanf("%d", &n);
    int a[n][n];
    for (int i = 0; i < n; ++i){
        for (int j = 0; j < n; ++j){
            scanf("%d", &a[i][j]);
        }
    }

    int foatlo = 0, mellekatlo = 0;
    for (int i = 0; i < n; ++i) {
        foatlo += a[i][i];
        mellekatlo += a[i][n-1-i];
    }
    printf("FőátlóK összege = %d\nMellékátlók összege = %d\n", foatlo,
mellekatlo );
    return 0;
}
```

Komplex feladatok Két mátrix összeszorzása

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 100
// Függvény a mátrix kiírásához
void printMatrix(int matrix[MAX][MAX], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }
}

// Függvény a két mátrix szorzatának kiszámításához

void multiplyMatrices(int matrix1[MAX][MAX], int rows1, int cols1, int
matrix2[MAX][MAX], int rows2, int cols2, int result[MAX][MAX]) {
    // Ellenőrzés: a két mátrix szorzatának megfelelő mérete
    if (cols1 != rows2) {
        printf("A ket matrix szorzatanak merete nem megfelelo.\n");
        exit(1); // Kilépés hibakóddal
    }

    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols2; j++) {
            result[i][j] = 0;

            for (int k = 0; k < cols1; k++) {
                result[i][j] += matrix1[i][k] * matrix2[k][j];
            }
        }
    }
}

int main() {
    // Véletlenszám inicializálása
    srand(time(NULL));

    // Mátrix méreteinek bekérése
    int rows1, cols1, rows2, cols2;

    printf("Adja meg az elso matrix meretet (sor oszlop): ");
    scanf("%d %d", &rows1, &cols1); //C99 előtt két sorba kell írni, két
scanf (codeblock)

    printf("Adja meg a masodik matrix meretet (sor oszlop): ");
```

```
    scanf("%d %d", &rows2, &cols2); //C99 előtt két sorba kell írni, két
scanf (codeblock)


    if (rows1 > MAX || cols1 > MAX || rows2 > MAX || cols2 > MAX) {
        printf("A megadott méret túl nagy. Maximum méret: %d\n", MAX);
        return 1; // Kilépés hibakóddal
    }

    // Mátrixok létrehozása és inicializálása véletlenszámokkal
    int matrix1[MAX][MAX], matrix2[MAX][MAX], result[MAX][MAX];

    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols1; j++) {
            matrix1[i][j] = rand() % 100; // Véletlenszám 0 és 99 között
        }
    }

    for (int i = 0; i < rows2; i++) {
        for (int j = 0; j < cols2; j++) {
            matrix2[i][j] = rand() % 100; // Véletlenszám 0 és 99 között
        }
    }

    // Eredeti mátrixok kiírása
    printf("Eredeti elso matrix:\n");
    printMatrix(matrix1, rows1, cols1);
    printf("\nEredeti masodik matrix:\n");
    printMatrix(matrix2, rows2, cols2);

    // Mátrixok szorzata
    multiplyMatrices(matrix1, rows1, cols1, matrix2, rows2, cols2, result);

    // Szorzat kiírása
    printf("\nA ket matrix szorzata:\n");
    printMatrix(result, rows1, cols2);

    return 0; // Sikeres végrehajtás
}
```

Mátrix maximális részmátrix megtalálása

```
#include <stdio.h>
#include <stdlib.h>
#define MERET 100
int matrix[MERET + 1][MERET + 1];
int main() {
    int n, bal, felso, jobb, also, i, j;
    long max;
    scanf("%d", &n);
```

```
    for (i = 0; i < n; i++){
        for (j = 0; j < n; j++){
            scanf("%d", &matrix[i][j]);
        }
    }
    max = matrix[0][0];
    for (bal = 0; bal < n; bal++){
        for (felso = 0; felso < n; felso++){
            for (jobb = bal; jobb < n; jobb++){
                for (also = felso; also < n; also++) {
                    long osszeg = 0;
                    for (i = bal; i <= jobb; i++){
                        for (j = felso; j <= also; j++){
                            osszeg += matrix[i][j];
                        }
                    }
                    if (osszeg > max){
                        max = osszeg;
                    }
                 }
             }
        }
    }
    printf("%ld\n", max);
}
```