Mátrix bevezetés

1D tömb deklarálása:

In [7]:

```
#include <stdio.h>

void output_array (int x, int a[]) {
    for (int i = 0; i < x; i++) {
        printf("%4d", a[i]);
    }
    printf("\n");
}

int main() {
    int N = 5;
    int array_1[5] = {1,2,3,4,5};
    int array_2[] = {1,2,3,4,5};

    output_array(N, array_1);
    printf("\n");
    output_array(N, array_2);
}
```

```
   1   2   3   4   5

   1   2   3   4   5
```

2D tömb deklarálása:

In [13]:

```
#include <stdio.h>

void output_2d_matrix (int row, int column, int a[row][column]) {
    printf("\n");
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < column; j++) {
            printf("%d ", a[i][j]);
        }
    printf("\n");
    }
}

int main() {
    int N = 5, M = 2;
    int array_1[5][2] = {{1,2},
                         {3,4},
                         {5,4},
                         {1,2},
                         {1,3}};
    int array_2[][2] = {{1,2},{3,4},{5,4},{1,2},{1,3}};
    output_2d_matrix(N, M, array_1);
    output_2d_matrix(N, M, array_2);
}
```

```
1 2
3 4
5 4
1 2
1 3

1 2
3 4
```

```
5 4
1 2
1 3
```

3D tömb deklarálása:

In [31]:

```
#include <stdio.h>

void output_3d_matrix (int x, int y, int z, int a[x][y][z]) {
    for (int i = 0; i < x; i++) {
        for (int j = 0; j < y; j++) {
            for (int k = 0; k < z; k++) {
            printf("%d ", a[i][j][k]);
            }
        printf("\n");
        }
    printf("\n");
    }
printf("\n");
}

int main() {
    int N = 5, M = 2, O = 3;
    int array_1[5][2][3] = {{{1,2,3},{3,4,5}},
                            {{5,4,3},{1,2,3}},
                            {{1,3,2},{1,2,3}},
                            {{6,7,8},{8,9,5}},
                            {{4,3,2},{1,2,3}}};
    int array_2[][2][3] = {{{1,2,3},{3,4,5}},{{5,4,3},{1,2,3}},{{1,3,2},{1,2,3}},{{6,7,8}
,{8,9,5}},{{4,3,2},{1,2,3}}};
    output_3d_matrix(N, M, O, array_1);
    output_3d_matrix(N, M, O, array_2);
}
```

```
1 2 3
3 4 5

5 4 3
1 2 3

1 3 2
1 2 3

6 7 8
8 9 5

4 3 2
1 2 3


1 2 3
3 4 5

5 4 3
1 2 3

1 3 2
1 2 3

6 7 8
8 9 5

4 3 2
1 2 3
```

2D mátrix rendezése soronként

In [21]:

```c
#include <stdio.h>

void output_matrix(int sor, int oszlop, int tomb[sor][oszlop]) {
    for (int i = 0; i < sor; i++) {
        printf("\n%d. sor:\t",i+1);
        for (int j = 0; j < oszlop; j++) {
            printf("%3d ",tomb[i][j]);
        }
    }
}

void gen_matrix(int sor, int oszlop, int tomb[sor][oszlop]) {
    for (int i = 0; i < sor; i++) {
        for (int j = 0; j < oszlop; j++) {
            tomb[i][j] = rand() % 10 + 1;
        }
    }
}

void sort_rows_matrix(int sor, int oszlop, int tomb[sor][oszlop]) {
    int temp = 0;
    for (int i = 0; i < sor; i++) {
        for (int j = 0; j < oszlop; j++) {
            for (int k = j + 1; k < oszlop; k++) {
                if (tomb[i][j] > tomb[i][k]) {
                    temp = tomb[i][j];
                    tomb[i][j] = tomb[i][k];
                    tomb[i][k] = temp;
                }
            }
        }
    }
}

int main() {
    int array_3[5][5];

    gen_matrix(5,5,array_3);
    printf("\n\nRendezetlen:");
    output_matrix(5,5,array_3);
    sort_rows_matrix(5,5,array_3);
    printf("\n\nRendezett:");
    output_matrix(5,5,array_3);
}
```

```
Rendezetlen:
1. sor:    4   7   8   6   4
2. sor:    6   7   3  10   2
3. sor:    3   8   1  10   4
4. sor:    7   1   7   3   7
5. sor:    2   9   8  10   3

Rendezett:
1. sor:    4   4   6   7   8
2. sor:    2   3   6   7  10
3. sor:    1   3   4   8  10
4. sor:    1   3   7   7   7
5. sor:    2   3   8   9  10
```

2D mátrix rendezése

In [1]:

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

```
void output_matrix(int sor, int oszlop, int tomb[sor][oszlop]) {
    for (int i = 0; i < sor; i++) {
        printf("\n%d. sor:\t",i+1);
        for (int j = 0; j < oszlop; j++) {
            printf("%3d ",tomb[i][j]);
        }
    }
}

void gen_matrix(int sor, int oszlop, int tomb[sor][oszlop]) {
    for (int i = 0; i < sor; i++) {
        for (int j = 0; j < oszlop; j++) {
            tomb[i][j] = rand() % 10 + 1;
        }
    }
}

void sort_matrix(int n, int m, int x[n][m]) {
    int temp = 0, l;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            l = j + 1;
            for (int k = i; k < n; k++) {
                while (l < m) {
                    if (x[i][j] > x[k][l]) {
                        temp = x[k][l];
                        x[k][l] = x[i][j];
                        x[i][j] = temp;
                    }
                l++;
                }
            l = 0;
            }
        }
    }
}

int main() {
    int array_3[5][5];

    gen_matrix(5,5,array_3);
    printf("\n\nRendezetlen:");
    output_matrix(5,5,array_3);
    sort_matrix(5,5,array_3);
    printf("\n\nRendezett:");
    output_matrix(5,5,array_3);
}
```

```
Rendezetlen:
1. sor:   4   7   8   6   4
2. sor:   6   7   3  10   2
3. sor:   3   8   1  10   4
4. sor:   7   1   7   3   7
5. sor:   2   9   8  10   3

Rendezett:
1. sor:   1   1   2   2   3
2. sor:   3   3   3   4   4
3. sor:   4   6   6   7   7
4. sor:   7   7   7   8   8
5. sor:   8   9  10  10  10
```

Feladatok

1. **Rendezze a teljes tömböt csökkenő sorrendbe!**
2. **Rendezze a tömb oszlopait növekvő sorrendbe!**
3. **A felhasználó által kiválasztott elem értékét írja át 0-ra!**
4. **A felhasználó által kiválasztott sor, vagy oszlop elemeinek értékét írja át 0-ra!**

In [ ]: