Matrice. Operaţii cu matrice: adunare, înmulţire. Reprezentarea în memorie.

Matrice

Matricea este o colecţie omogenă şi bidimensională de elemente.
Acestea pot fi accesate prin intermediul a doi indici, numerotaţi, ca şi în cazul vectorilor, începand de la 0.
Declaraţia unei matrice este de forma:

```
<tip_elemente> <nume_matrice>[<dim_1>][<dim_2>];

int mat[5][10];

#define MAX_ELEM 100
float a[MAX_ELEM][MAX_ELEM];
```

Numărul de elemente ale unei matrice va fi:

```
dim1_1 * dim_2
```

Tablouri multidimensionale

```
<tip_elemente> <nume_tablou>[<dim_1>][<dim_2
    >]...[<dim_n>];

int cube[3][3][3];
```

Reprezentarea in memorie

Standardul C impune ca un tablou să fie memorat într-o zonă continuă de memorie, astfel ca pentru un tabloul de forma:

```
T tab[dim1][dim2]...[dimn];
```

Dimensiunea ocupată în memorie va fi:

```
sizeof(T)*dim1*dim2*...*dimn
```

Vom considera în continuare cazul particular al unui vector vect de lungime n, şi al unui element oarecare al acestuia, de pe pozitia i.
Atunci când întalneşte numele vect, compilatorul va intelege “adresa în memorie de la care începe vectorul vect”.

```
char a[100];
a[0] = 1;
3[a] = 5;
```

Asignează 5 variabilei de tip char de la adresa:

```
3 + a * sizeof(char) = 3 + a
```

Este echivalentă cu:

```
a[3] = 5;
```

Exemple de programre

Declararea unei matrici unitate:

```
#define M 20 /* nr maxim de linii si de coloane */

int main() {
  float unit[M][M];
  int i,j,n;

  printf("nr.linii/coloane:␣");
  scanf("%d", &n);
  if (n > M) {
    return;
  }

  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
      if (i != j) {
        unit[i][j]=0;
      } else {
        unit[i][j]=1;
      }
    }
  }

  return 0;
}
```

Citire/scriere de matrice de reali:

```
int main() {
  int nl, nc, i, j;
  float a[20][20];
   /* Citire de matrice */
  printf("nr.linii:␣");
  scanf("%d", &nl);
  printf("nr.coloane:␣");
  scanf("%d", &nc);
  if (nl > 20 || nc > 20) {
    printf("Eroare:␣dimensiuni␣>␣20␣\n");
    return ;
  }

  for (i = 0; i < nl; i++) {
    for (j = 0; j < nc; j++) {
      scanf("%f", &a[i][j]);
    }
  }
  /* Afisare matrice */
  for (i = 0; i < nl; i++) {
    for (j = 0; j < nc; j++) {
      printf("%f␣",a[i][j]);
    }
    printf ("\n");
  }
  return 0;
}
```

Transpunerea unei matrici patratice

```
#include <stdio.h>

#define N 100


void transpose_matrix(int m[N][N], int n)
{
    int i, j, tmp;

    for (i = 0; i < n; i++) {
        for (j = i + 1; j < n; j++) {
            tmp = m[i][j];
            m[i][j] = m[j][i];
            m[j][i] = tmp;
        }
    }
}


void print_matrix(int m[N][N], int n)
{
    int i, j;

    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d␣", m[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}


int main(void)
{
    int i, n = 3;

    int V[N][N] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    print_matrix(V, n);
    transpose_matrix(V, n);
    print_matrix(V, n);

    return 0;
}
```