Mátrixok, mutatók

Kormányos Andor

Komplex Rendszerek Fizikája Tanszék

2019. október 7.

Többdimenziós tömbök

Többdimenziós tömbök deklarációja hasonló a vektoroknál látotthoz:

```
1  tipus tombnev[meret1][meret2]...[meretn]
```

Továbbiakban csak kétdimenziós tömböket tekintünk.

```
1  tipus tombnev[meret1][meret2]
```

`meret1`: sorok száma
`meret2`: oszlopok száma

Többdimenziós tömbök

Többdimenziós tömbök deklarációja hasonló a vektoroknál látotthoz:

```
1  tipus tombnev[meret1][meret2]...[meretn]
```

Továbbiakban csak kétdimenziós tömböket tekintünk.

```
1  tipus tombnev[meret1][meret2]
```

`meret1`: sorok száma
`meret2`: oszlopok száma

Figyelem!

- `tombnev[meret1,meret2]` jelölés nem működik
- C-ben at elemek soronként haladva tárolódnak, fentről lefelé (row-major)
- ellentétben pl a FORTRAN-nal, ahol oszlopontként haladva, balról jobbra (column-major)

Mátrixok tárolása

Példa konstans mátrix megadására:

```
1  double matrix[3][2]={ {2.0, 4.1}, \\ 0.   sor
2                        {1.0, 4.0},  \\ 1.  sor
3                        {6.4, 9.3},  \\ 2.  sor
4                        }
```

Mátrixok tárolása

Példa konstans mátrix megadására:

```
1  double matrix[3][2]={ {2.0, 4.1}, \\ 0.   sor
2                        {1.0, 4.0},  \\ 1.  sor
3                        {6.4, 9.3},  \\ 2.  sor
4                        }
```

A mátrixok bejárása két `for` ciklussal történhet:

```
1
2   int main()
3
4   int i,j;
5   double matrix[3][2]={  {2.0, 4.1}, \\ 0.   sor
6                          {1.0, 4.0},  \\ 1.  sor
7                          {6.4, 9.3},  \\ 2.  sor
8                       }
9
10    for(i=0; i<3; i++)
11      {
12      for(j=0; j<2; j++)
13        {
14        printf("%d %d %f",i,j,matrix[i][j]);
15        }
16        printf("\n");
17      }
```

A második index (oszlop) számozása is 0-val kezdődik!

Mátrix tárolása vektorban

Adott egy $M \times N$ méretű mátrix, amit tárolni szeretnénk. Ehhez gyakran kényelmes vektorokat és mutatókat használni.

Adattípus definiálásához szükséges:

- két változó a méret tárolására: `cols`, `rows`
- egy $M \times N$ `double` tárolására elegendő memóriaterület
- a mátrixot tároló memória területre mutató `double *mat` pointer
- indexeljük a mátrix sorait az `i`, oszlopait a `j` változókkal

Mátrix tárolása vektorban

Adott egy $M \times N$ méretű mátrix, amit tárolni szeretnénk. Ehhez gyakran kényelmes vektorokat és mutatókat használni.

Adattípus definiálásához szükséges:

- két változó a méret tárolására: `cols`, `rows`
- egy $M \times N$ `double` tárolására elegendő memóriaterület
- a mátrixot tároló memória területre mutató `double *mat` pointer
- indexeljük a mátrix sorait az `i`, oszlopait a `j` változókkal

Hogyan kapható meg `i`-ből, `j`-ből, és a mátrix méreteiből,
hogy mit kell `mat[ ... ]` indexelésekor a zárójelek közé írni?

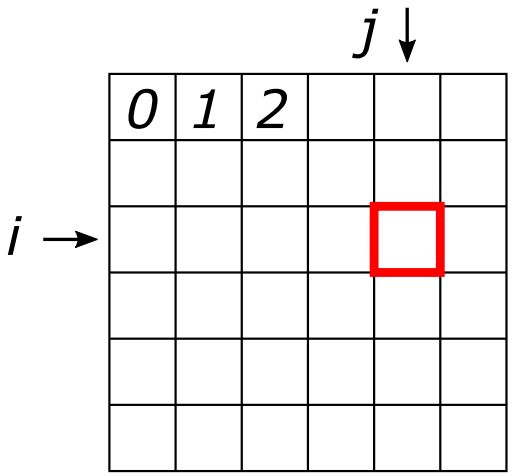
Mátrix tárolása vektorban

Adott egy $M \times N$ méretű mátrix, amit tárolni szeretnénk. Ehhez gyakran kényelmes vektorokat és mutatókat használni.

Adattípus definiálásához szükséges:

- két változó a méret tárolására: `cols`, `rows`
- egy $M \times N$ `double` tárolására elegendő memóriaterület
- a mátrixot tároló memória területre mutató `double *mat` pointer
- indexeljük a mátrix sorait az `i`, oszlopait a `j` változókkal

Hogyan kapható meg `i`-ből, `j`-ből, és a mátrix méreteiből,
hogy mit kell `mat[ ... ]` indexelésekor a zárójelek közé írni?



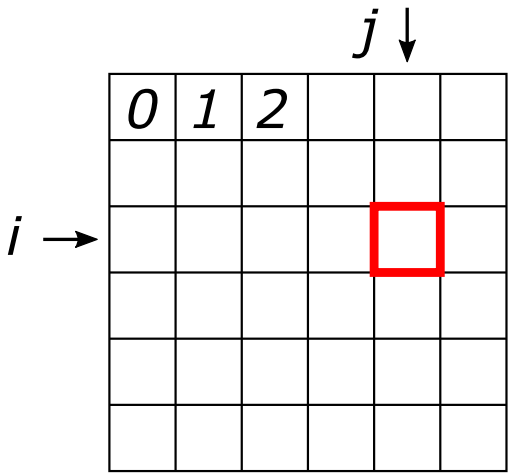
Mátrix tárolása vektorban

Adott egy $M \times N$ méretű mátrix, amit tárolni szeretnénk. Ehhez gyakran kényelmes vektorokat és mutatókat használni.

Adattípus definiálásához szükséges:

- két változó a méret tárolására: `cols`, `rows`
- egy $M \times N$ `double` tárolására elegendő memóriaterület
- a mátrixot tároló memória területre mutató `double *mat` pointer
- indexeljük a mátrix sorait az `i`, oszlopait a `j` változókkal

Hogyan kapható meg `i`-ből, `j`-ből, és a mátrix méreteiből, hogy mit kell `mat[ ... ]` indexelésekor a zárójelek közé írni?



A mátrix soronkénti (row-major) tárolása esetén az i,j elem:

```
m[i * cols + j]
```

Pédafüggvények mátrixok kezelésére

A vektorok kezelésére használt függvényekhez hasonlóan:

```
1  double *alloc_matrix(int cols, int rows) {
2    double *matr = (double*)malloc(cols * rows * sizeof(double));
3    if (matr == 0) {
4      printf("Memory allocation error.\n");
5      exit(-1);
6    }
7    return matr;
8  }
9
10 void read_matrix(FILE* f, double *m, int cols, int rows) {
11   for (int i = 0; i < rows; i++) {
12     for (int j = 0; j < cols; j++) {
13       fscanf(f, "%lf", &m[i * cols + j]);
14     }
15   }
16 }
17
18 void write_matrix(FILE* f, double  *m, int cols, int rows) {
19   for (int i = 0; i < rows; i++) {
20     for (int j = 0; j < cols; j++) {
21       fprintf(f, "%f ", m[i * cols + j]);
22     }
23     fprintf(f, "\n");
24   }
25 }
```