Egitura dinamikoen erabilera

Jarraian adierazten diren ekintzak beteko dituen programa bat egin behar da:

1. Lehenengo teklatutik irakurriko da matrizeari buruzko informazioa duen **fitxategiaren izena** (Fitxategi horretan etorriko dira errenkada eta zutabe kopurua; baita matrizea osatzen duten balio desberdinak ere (denak **double** motakoak). Kontutan izan fitxategiaren edukia **ez** datorrela **karaktere** formatuan).
2. Fitxategi horretatik irakurriko dira errenkada (**m**) eta zutabe (**n**) kopurua.
3. Jarraian ***mxn*** matrize bat sortuko da ***memoriaren erreserba dinamiko*** bidez.
4. Hasierako fitxatgitik irakurriko dira matrizearen elementuen balioak (balioak ***double*** motakoak izango dira).
5. Matrize pantailan bistaratu
6. Programak erabiltzaileari galdetuko dio ***zein errenkada eta zein zutabe ezabatu*** nahi dituen. Erabiltzaileak datu hauek teklatutik sartuko ditu. Adierazitako zutabe eta errenkadak benetan existitzen direla ziurtatu behar da (adibidez, 5x3 matrize batean ezin da -1 errenkada ezabatu, ezta 4 zutabea ere).
7. ***(m-1)x(n-1) tamainako beste matrize bat*** sortuko da, ***baita memoriaren erreserba dinamikoa erabilita***. Matrize honetan hasierako matrizearen elementuak kopiatuko dira, ezabatu nahi diren errenkada eta zutabekoak izan ezik.
8. Azkenik, ***bi matrizeak pantailatik bistaratuko*** dira, hasierakoa eta errenkada eta zutabea ezabatu ondoren gelditzen dena.

Ariketa hau egiteko ***funtzioak*** erabili behar dira.

Gainera ariketa egiteko ***tMatriz*** izeneko datu mota definitu behar da. **tMatriz** datu egitura bat izango da, eta bere osagaiak izango dira: matrizearen errenkada kopurua, zutabe kopurua eta matrizearen edukia.

```
typedef struct Matriz{
       int f;
       int c;
      double **m;
}tMatriz;
```

```
/*
 * Funcion  initMatriz()
 *
 * Parametros de entrada:
 *       int m: numero de filas de la matriz
 *       int n: numero de columnas de la matriz
 *
 * Parametros de salida:
 *       Devuelve la direccion del registro de la matriz
 *
 * Descripcion:
 *       guarda en el registro las dos dimensiones del array, realiza la
 *       reserva de memoria y guarda la direccion de dicha memoria
 */
tMatriz * initMatriz (int m, int n);
```

```
/*
 * Funcion  cargarMatriz()
 *
 * Parametros de entrada:
 *       char *nomFich: nombre del fichero de datos
 *
 * Parametros de salida:
 *       Devuelve la direccion del registro de la matriz con los datos
 *
 * Descripcion:
 *       lee del fichero de datos las dimensiones, con los cuales inicializa
 *       el registro para a continuacion seguir leyendo y guardando en la
 *       memoria reservada dichos datos.
 */
tMatriz * cargarMatriz (char *nomFich);
```

```
/*
 * Funcion  reducirMatriz()
 *
 * Parametros de entrada:
 *       tMatriz *mat: direccion con los datos de la matriz
 *       int fil: la fila a eliminar
 *       int col: la columna a eliminar
 *
 * Parametros de salida:
 *       Devuelve la direccion del registro con la matriz reducida
 *
 * Descripcion:
 *       crea una nueva matriz sin la fila y columna correspondiente
 */
tMatriz * reducirMatriz(tMatriz *mat, int fil, int col);
```

```
/*
 * Funcion  imprimirMatriz()
 *
 * Parametros de entrada:
 *       tMatriz *mat: direccion con los datos de la matriz
 *
 * Parametros de salida:
 *       Ninguno
 *
 * Descripcion:
 *       lee del fichero de datos las dimensiones, con los cuales inicializa
 *       el registro para a continuacion seguir leyendo y guardando en la
 *       memoria reservada dichos datos.
 */
void imprimirMatriz(tMatriz *mat);
```

```
/*
 * Funcion  liberarMemoria()
 *
 * Parametros de entrada:
 *       tMatriz *mat: direccion con los datos de la matriz
 *
 * Parametros de salida:
 *       Ninguno
 *
 * Descripcion:
 *       libera la memoria utilizada por el array al que hace referencia el registro
que recibe la función.
*/
void liberarMemoria (tMatriz *mat);
```
