Fejlesztői dokumentáció

Balog Ádám Márk (ELAO0E)

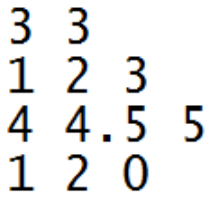
December 2019

1. Függvénykönyvtár használata

A felhasználó az általa írt programban meghívja a függvénykönyvtárat a következő sorral a fejlécben:

#include "matrix.h"

Ezután használhatja az alább kifejtett függvények bármelyikét, amennyiben az adott fájl mely a mátrixot tartalmazza a következő struktúrájú:



1. ábra. Példa egy 3x3-as mátrix bemenetre

Első sor 2 egész szám, szóközzel elválasztva. Ez írja le a mátrix méretét, első a sorok, második az oszlopok számát. tartalmazza.
Ezután következnek a mátrix elemei. Sorban az új elemet szóközzel, új oszlopot sortöréssel választjuk el. Tizedestörteket '.'-tal jelöljük. A függvények kivétel nélkül olyan pointerre mutató pointerrel operálnak, melyek a mátrix elemein kívül tartalmazzák annak méretét is a képen látható elrendezéshez hasonlóan. Így tehát az indexelés a valódi értékek elérése esetében 'oszlop'=1 és 'sor'=0 -val kezdődik.
Példa: A fenti mátrix esetében a matematikailag (1,1) helyen álló elem elérése:

$$M[1][0] = 1 \tag{1}$$

2. Függvények

double* soronkent(char* sor)

Paraméterek:

- **char*** sor: karaktertömbre mutató pointer

Változók:

- **int** space: szóközöket számoló változó
- **int** i: ciklusváltozó
- **int** j: ciklusváltozó
- **int** szamlalo: ciklusváltozó
- **int** elozo: ciklusváltozó
- **char*** szamma: karakterpointer, később ezt alakítja számmá
- **double*** elsosor: eredmény pointer, a karaktertömbből számmá alakított értékeket tartalmazza

Működése:

Megszámolja az adott stringben található szóközöket. Ennek függvényében megfelelő méretű memóriát foglal 'elsosor'-nak. Először az első szóközig beolvassa az adatot, azt számmá konvertálja és beírja 'elsosor' megfelelő címébe. Megjegyzi az előző szóköz helyzetét, így az algoritmust előről kezdve a következő számot írja 'elsosor'-ba, egészen a string végéig. Végül visszatér az adott double tömbbel.

double** beolvas(char* hely)

Paraméterek:

- **char*** hely: fájl elérési helye

Változók:

- **char** szoveg[1000]: 1000 karakter hosszúságig beolvassa a fájlból az adott sort
- **double*** sorolv: 'szoveg'-ből készített double tömb, mely a mátrix sora lesz
- **double**** matrix: végeredmény mátrix, pointerre mutató pointer
- **int** sor: sorszám

- **int** oszlop: oszlopszám
- **int** i: ciklusváltozó

Működése:

Megnyitja az adott szövegfájlt. Az első sorból, mely a mátrix méretét tartalmazza, átállítja ennek megfelelően 'sor' és 'oszlop' értékeit. Ezután ezek függvényében ismétli az algoritmust, csak már a valódi mátrixelemeket tartalmazó sorokra, és az így készített double tömböket a megfelelő 'matrix' címhez társítja. A végén visszatér a beolvasott mátrixra mutató pointerrel.

void cout(double** matrix)

Paraméterek:

- **double**** matrix: pointerre mutató pointer, a konzolra kiírandó mátrix

Változók:

- **int** sori: sorban futó index
- **int** oszlopi: oszlopban futó index
- **int** sor: paramétermátrix sorainak hossza
- **int** oszlop: paramétermátrix oszlopainak hossza

Működése:

A mátrix elemeit egyesével, 4 szóközzel elválasztva, sorokat 2 sortöréssel tagolva kiírja a konzolablakba. A méretét (azaz a 0. sort) nem írja ki.

double** sum(double** M1, double** M2)

Paraméterek:

- **double**** M1: pointerre mutató pointer, az egyik összeadandó mátrix
- **double**** M2: pointerre mutató pointer, a másik összeadandó mátrix

Változók:

- **int** sori: sorban futó index
- **int** oszlopi: oszlopban futó index
- **double** c: köztes változó, két mátrix megfelelő elemeinek összege, ez lesz a összegmátrix eleme
- **double**** osszeg: eredmény mátrix

Működése:

Nem azonos "alakú" mátrixok esetén hibát dob(1). Ellenkező esetben elemenként feltölti az eredemény mátrixot a két paramétermátrix megfelelő elemeinek összegével, majd visszatért az eredeménymátrixra mutató pointerrel.

double** sub(double** M1, double** M2)

Paraméterek:

- **double**** M1: pointerre mutató pointer, a kisebbítendő mátrix
- **double**** M2: pointerre mutató pointer, a kivonandó mátrix

Változók:

- **int** sori: sorban futó index
- **int** oszlopi: oszlopban futó index
- **double** a: köztes változó, két mátrix megfelelő elemeinek különbsége, ez lesz a különbségmátrix eleme
- **double** **kul: eredmény mátrix

Működése:

Nem azonos "alakú" mátrixok esetén hibát dob(1). Ellenkező esetben elemenként feltölti az eredemény mátrixot a két paramétermátrix megfelelő elemeinek különbségével, majd visszatért az eredeménymátrixra mutató pointerrel.

void mulc(double** M, double lambda)

Paraméterek:

- **double**** M: pointerre mutató pointer, a konstanssal szorzandó mátrix
- **double**lambda: a konstans amivel szorzunk

Változók:

- **int** sori: sorban futó index
- **int** oszlopi: oszlopban futó index

Működése:

Végigmegy a paraméterben megadott mátrix osszes elemén és beszorozza az adott lambda konstanssal.

double** mult(double** M1, double** M2)

Paraméterek:

- **double**** M1: pointerre mutató pointer, a szorzásban bal oldali mátrix
- **double**** M2: pointerre mutató pointer, a szorzásban jobb oldali mátrix

Változók:

- **int** sorelem: sorban futó index
- **int** oszlop: oszlopban futó index
- **double** a: köztes változó, két mátrix megfelelő elemeinek szorzata itt adódik össze, ez lesz a szorzatmátrix eleme
- **int** j: ciklusváltozó
- **double**** result: eredmény mátrix

Működése:

Nem megfelelő "alakú" mátrixok esetén hibát dob. Egyébként lefoglal az eredménymátrixnak megfelelő méretű tömböt és a mátrixszorzás szabályának megfelelően ciklusok révén feltölti az eredmény mátrixot, majd visszaadja a rá mutató pointert.

void freem(double** M)

Paraméterek:

- **double**** M: pointerre mutató pointer, a mátrix aminek a helyét fel szeretnénk szabadítani

Változók:

- **int** sor: mátrix sorszáma
- **int** i: ciklusváltozó

Működése:

Először a valódi értékkel bíró sorokat szabadítja fel, majd a mátrix méretét tartalmazó 0. sort is.

double det(double** A)

Paraméterek:

- **double**** A: pointerre mutató pointer, a mátrix melynek determinánsát szeretnénk tudni

Változók:

- **int** i: ciklusváltozó
- **double** determinant: leendő determináns
- **double**** atmeneti: átmeneti mátrix mely csak a rekurzív determinánsszámítás köztes almátrixainak tárolására való

Működése:

Ha a mátrix nem négyzetes, hibát dob, és kilép(1).Ellenkező esetben ha a mátrix 1x1-es, visszatér az adott elemmel, ha nxn-es (n¿=2), a felső sorra alkalmazott kifejtési szabály szerint rekurzívan kiszámolja a determinánst. Közben az átmeneti mátrixokat sorra felszabadítja. A végén visszatér az adott valós számmal.

double** almatrix(double** M, int oszlop, int sor)

Paraméterek:

- **double**** M: pointerre mutató pointer, a csonkítandó mátrix
- **int** oszlop, sor: oszlopindex, sorindex, ezeket vágja ki a mátrixból

Változók:

- **double** ujsorszam: új mátrix sorszáma
- **double** ujoszlopszam: új mátrix oszlopszáma
- **int** i,j,k: ciklusváltozók
- **double**** ujmatrix: eredmény mátrix pointere

Működése:

Lefoglal a memóriában egy megfelelő formátumú és méretű blokkot a mátrixnak, majd feltölti az eredeti mátrix azon elemeivel megfelelő sorrendben, melyekre igaz, hogy oszlopindexük nem egyenlő a paraméterben megadott 'oszlop'-pal és sorindexük nem egyenlő az ugyanitt megadott 'sor'-ral. A végén visszatér a csonkított mátrixra mutató pointerrel.

double** kulonsorra(double** M, int sor)

Paraméterek:

- **double**** M: pointerre mutató pointer, a mátrix melyből ki akarunk szervezni egy adott sort
- **int** sor: kiszervezendő sor indexe

Változók:

- **double**** sorvektor: függvények általi kezelésre alkalmas formátumú, a mátrix kiszervezendő sorát tartalmazó 'mátrix'(sorvektor)
- **int** i: ciklusváltozó

Működése:

Lefoglal a memóriában egy megfelelő formátumú és méretű blokkot a mátrixnak, majd a kiszervezendő sor elemeit belepakolja, és visszatér az eredményre mutató pointerre.

double** transponalt(double** M)

Paraméterek:

- **double**** M: pointerre mutató pointer, a transzponálandó mátrix

Változók:

- **int** i,j: ciklusváltozók

Működése:

Lefoglal a memóriában egy megfelelő formátumú és méretű blokkot a mátrixnak, majd ciklusok révén transzponálja a mátrixot az elemek megfelelő helyre való írásával. Végén visszaadja az eredménymátrixra mutató pointert.

void sorcsere(double** M, int i, int j)

Paraméterek:

- **double**** M: pointerre mutató pointer, a cserélendő sorokat tartalmazó pointer
- **int** i, j: sor és oszlopszám

Változók:

- **double*** koztes: ideiglenes pointer, mely rövid ideig a cserélendő sor értékével bír

Működése:

i-edik sort ideiglenesen a 'koztes' pointerbe teszi, majd felülírja a j-edik sorral. Végül a j-edik sort felülírja a koztes sorral, azaz az eredeti i-edik sorral. A végén felszabadítja a koztes memóriáját.

double** adj(double** M)

Paraméterek:

- **double**** M: pointerre mutató pointer, mátrix melynek az aldeterminánsaiból álló mátrixra vagyunk kíváncsiak

Változók:

- **int** i,j: ciklusváltozók
- **double**** inverse: leendő adjungált mátrix
- **double** ertek: köztes változó, az adjungált mátrix elemeinek beírására szolgál

Működése:

Nem négyzetes mátrix esetén kilép (1). Lefoglal egy megfelelő méretű tömböt a memóriában az adjungált mátrixnak. Ezután feltölti elemenként az eredeti mátrix megfelelő előjelű aldeterminánsaival. A végén visszatér az adjungált mátrixra mutató pointerrel.

double** inverse(double** M)

Paraméterek:

- **double**** M: pointerre mutató pointer, az invertálandó mátrix

Változók:

- **double** a: M determinánsának reciproka
- **double**** eredmeny: eredménymátrix pointere

Működése:

Kiszámítja a mátrix inverzét determinánsos módszerrel: determináns reciprokával beszorozza az adjungált mátrix transzponáltját, majd visszatér az eredménnyel.

void letisztaz(double** M)

Paraméterek:

- **double**** M: pointerre mutató pointer, a tisztázandó mátrix

Változók:

- **int** i,j: ciklusváltozók

Működése:

Kényelmi szerepe van, az eliminációk és műveletek során double érték korlátoltságából adódóan előfordulhat, hogy 0 helyett egy igen kicsi számot rak a mátrixba. Ezen függvény 1*10(-10)-nél kisebb 0-tól való eltérés esetén az adott értéket 0-ra állítja a mátrixban.

double** deepcpy(double** M)

Paraméterek:

- **double**** M: pointerre mutató pointer, a másolandó mátrix

Változók:

- **int** i,j: ciklusváltozók
- **double**** deep: végeredmény mátrixra mutató pointer

Működése:

Lefoglal a memóriában egy ugyan olyan formátumú és méretű blokkot a mátrixnak, mint a paraméterben megadott, majd a másolandó mátrix elemét egyenként beírja a megfelelő címbe. A végén visszatér a lemásolt mátrixra mutató pointerrel.

double** GJE(double** M)

Paraméterek:

- **double**** M: pointerre mutató pointer, a Gauss eliminálandó mátrix

Változók:

- **double**** eredmeny: eredmény mátrix tárolására szolgál, ezt adja vissza a függvény
- **double**** sorv: függvények számára kezelhető formátumú, valójában egy sort tartalmazó változó, mely végül az eredmény mátrix sora lesz
- **double**** kisebbitendo: függvények számára kezelhető formátumú, valójában egy sort tartalmazó változó, az adott változtatandó sort tartalmazza
- **double**** kivonando: függvények számára kezelhető formátumú, valójában egy sort tartalmazó változó, ennek alkalmas konstans-szorosai kerülnek kivonásra 'kisebbitendo'-ből
- **int** i,j,b: ciklusváltozók
- **double** a: adott sor konstans szorzója az eliminálás során

Működése:

Teljesen lemásolja a paraméterként megadott mátrixot, így az változatlan marad a folyamat végére. Először az első sorban megkeresi az első nem 0 elemet, majd a többi sorból kivonja ezen kiemelt sor megfelelő konstansszorosát, hogy a kiemelt sor első nem 0 eleme "alatt" (és felett) csak 0-k legyenek. Az algoritmust ismétli a 2., 3., stb. sorokra is. Amint az algoritmus befejeződött, az eredmény mátrixból már leolvasható a sortér egy bázisa, ám ezeket még soronként az első nem 0 elem szerint normálja. A végén visszatér a végeredmény mátrixra mutató pointerrel.

void Mmalloc(double** M,int i, int j)

Paraméterek:

- **double**** M: pointerre mutató pointer, a leendő mátrix
- **int** i, j: sor és oszlopszám

Változók:

- **int** k: ciklusváltozó

Működése:

Lefoglal a memóriában egy megfelelő formátumú és méretű blokkot a mátrixnak.

void fkiir(char* hely, double** mit)

Paraméterek:

- **char*** hely: a kimeneti fájl helye, neve (pl: c:\\users\\proba.txt)
- **double**** mit: az adott mátrixra mutató point

Változók:

- **int** oszlopszam: oszlopok számát tárolja
- **int** sorszam: sorok számát tárolja
- **int** i,j: indexváltozók

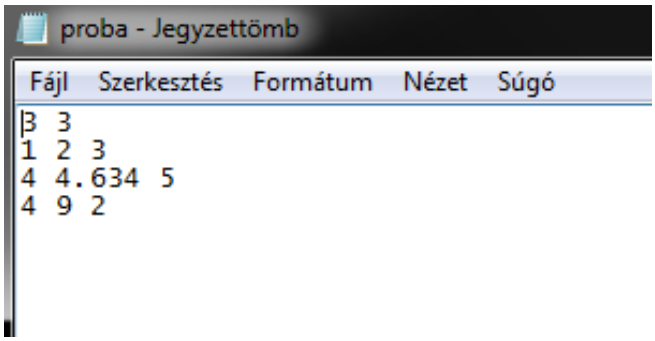
Működése:

Ciklusok által az adott fájlba kiírja a mátrixot olyan formában, ahogy a többi függvény azt használni tudja.

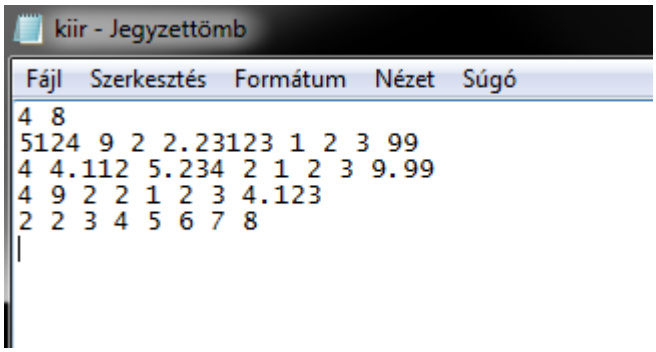
3. Fejlesztési lehetőségek

A függvénykönyvtár számos függvényén lehet még fejleszteni. Elsősorban a beolvasó függvény 1000 karakteres korlátoltságát lehetne javítani. Ezen kívül a különböző függvények "szebbé" tehetők, ha a köztes változók helyett az adott kifejezést írjuk paraméterbe, amivel kiszámoltuk a köztes változót.

4. Teszt:



2. ábra. beolvasott fájl struktúrája

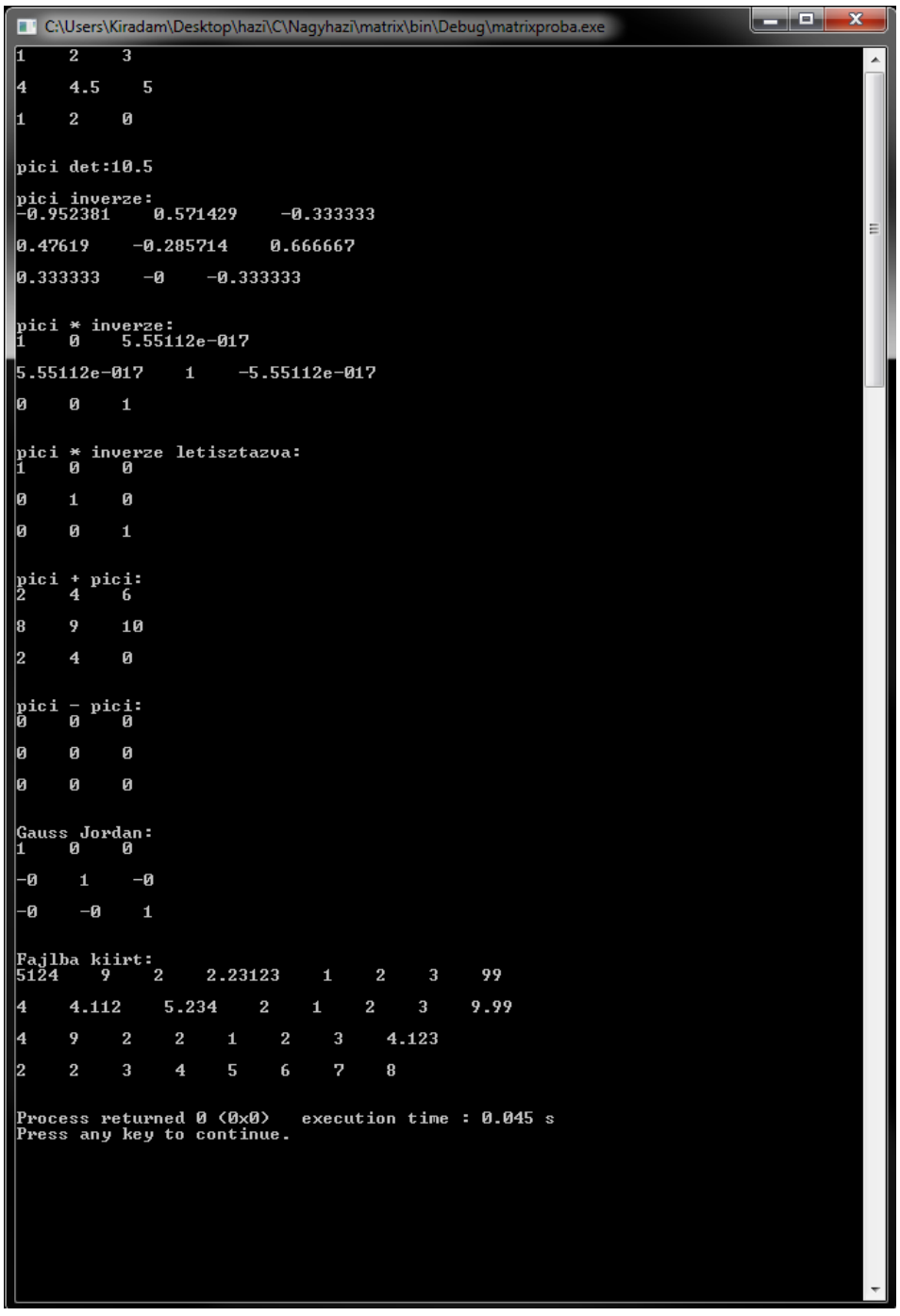


3. ábra. kiírt fájl struktúrája

```c
#include <stdio.h>
#include <stdlib.h>
#include "matrix.h"

int main()
{
    double** proba2=beolvas("c:\\Users\\Kiradam\\Desktop\\gyakorlas\\C\\proba2.txt");
    double** proba3=beolvas("c:\\Users\\Kiradam\\Desktop\\gyakorlas\\C\\proba3.txt");
    double** negyzetes=beolvas("c:\\Users\\Kiradam\\Desktop\\gyakorlas\\C\\negyzetes.txt");
    double** pici=beolvas("c:\\Users\\Kiradam\\Desktop\\gyakorlas\\C\\pici.txt");
    double** mindenes=mult(pici,inverse(pici));
    double** ujrabeolvas=beolvas("kiir.txt");
    cout(pici);
    printf("\npici det:%g\n",det(pici));
    printf("\npici inverze:\n");
    cout(inverse(pici));
    printf("\npici * inverze:\n");
    cout(mindenes);
    letisztaz(mindenes);
    printf("\npici * inverze letisztazva:\n");
    cout(mindenes);
    printf("\npici + pici:\n");
    cout(sum(pici,pici));
    printf("\npici - pici:\n");
    cout(sub(pici,pici));
    printf("\nGauss Jordan:\n");
    cout(GJE(pici));
    printf("\nFajlba kiirt:\n");
    fkiir("kiir.txt",proba2);
    cout(ujrabeolvas);
    freem(ujrabeolvas);
    freem(proba2);
    freem(proba3);
    freem(negyzetes);
    freem(pici);
    freem(mindenes);
    return 0;
}
```

4. ábra. tesztelő kód



5. ábra. tesztelő kód konzolos eredménye