Mátrix függvénykönyvtár programozói dokumentáció

Tartalomjegyzék

Mátrix függvénykönyvtár C-nyelven

A **Mátrix Függvénykönyvtár** célja a **mátrixműveletek** egyszerű és hatékony végrehajtása, a függvénykönyvtár a következő műveleteket képes végrehajtani:

- Két NxM-*es* mátrix összegének, illetve különbségének kiszámítása.
- Két NxM-*es* mátrix szorzatának kiszámítása.
- Gauss Elimináció lefuttatása NxN-*es* mátrixra.(Szabadtagok megadása esetén lineáris egyenletrendszer megoldása)
- Szabadtagok megadása esetén lineáris egyenletrendszer megoldása, ha a megoldás egyértelmű.
- NxN-*es* mátrix determinánsának kiszámítása.
- NxN-*es* mátrix transzponáltjának kiszámítása
- NxN-*es* mátrix inverzének a kiszámítása.
- NxN-*es* mátrix rangjának kiszámítása
- NxN-*es* mátrix nyomának kiszámítása
- Egy háromszög területének kiszámítása $3x3$-*as* mátrix determinánsának segítségével
- Három síkbeli pont kollinearitásának (egy egyenesen való elhelyezkedésének) eldöntése $3x3$-*as* mátrix determinánsának segítségével.

A programról általában

Az elkészített program a fentebb felsorolt műveleteket mind megvalósítja. A mátrixok beolvasása a felhasználó döntése alapján történik, vagy a standard bemenetről, vagy egy mátrixos művelet esetén ***bemenet_A.txt,*** két mátrixos művelet esetén a ***bemenet_A.txt,*** illetve ***bemenet_B.txt*** elnevezésű fileokból. A programot **`switch`** utasítások vezérlik. A modulokban azok a függvények szerepelnek, amelyekre a modul elnevezése is utal (például ***matrix_muveletk.c*** értelemszerűen a mátrixal való műveletfüggvényeket tartalmazza).

Minden sikeresen elvégzett művelet eredményét fileokba menti, de helytakarékosság miatt mindig csak az utoljára elvégzett művelet eredményei maradnak ténylegesen a file-ban. Mátrixot visszaadó műveletek esetén a ***kimenet_matrix.txt***-ba történik a mentés (összeadás, inverz stb.), egész vagy valós számérték esetén ***kimenet_eredmeny.txt***-ba (determináns, rang stb.).

A felhasznált adatszerkezet

A program egyetlen fő adatszerkezetet használ, amely egy mátrix adatait tárolja. A struktúra három alapvető adattaggal rendelkezik: a sorok és oszlopok száma, amelyet egész típusú változókban őriz és egy mutatóra mutató, amely dinamikus tömböket tárol, amelyek valós számokat tárolnak, azaz a mátrix tényleges elemeit.

```
typedef struct Matrix{
    int sorok;
    int oszlopok;
    double** adat;
} Matrix;
```

Mátrix létrehozásáért és dinamikus memóriakezelésért felelős függvények

- ***`Matrix* matrix_letrehozasa(int sorok, int oszlopok)`***

Értéket ad egy mátrix sorok és oszlop változóinak, majd dinamikusan memóriaterületet foglal a méreteknek megfelelően, ha a memóriafoglalás sikeres, akkor visszatér a létrehozott mátrixal.

- ***`void matrix_felszabaditasa(Matrix* matrix)`***

A függvény a dinamikusan foglalt memória felszabadítását végzi egy mátrix esetén, először a sorok elemeit, majd a sorokat tartalmazó tömböt, és végül magát a mátrixot szabadítja fel.

Mátrix beolvasásáért felelős függvények

- ***`Matrix* matrix_beolvas_billzet():`***

A függvény bekéri a felhasználótól a sorok és oszlopok számát, a megfelelő függvényhívással létrehoz egy mátrixot, majd bekéri a mátrix értékeit.

- ***`Matrix* matrix_beolvas_file(const char* filename):`***

A függvény tulajdonképpen az előzőhez hasonlóan működik, annyi különbséggel, hogy mind az oszlopok és sorok számát, mind az értékeket a paraméteren kapott nevű file-ból olvassa ki.

- ***Matrix* harom_pont_matrix():***

A függvény egy $3x3 - as$ mátrixot hoz létre, amelybe három pont koordinátáit olvassa be és utolsó oszlopát egyesekkel tölti fel (Kötött forma a háromszög területe és három pont kollinearitásának meghatározásához).

- ***int beolvasInt():***

Beolvas a standard bementről egy számot és egy karaktert, ellenőrzi, hogy érvényes-e a bemenet, ha igen visszatér vele, különben addig fut a függvény amíg nem kerül érvényes bemenet beolvasásra.

- ***double beolvasDouble():***

Beolvas a felhasználótól egy lebegőpontos számot a standard bemenetről. Sikeres beolvasás esetén visszaadja a lebegőpontos számot, egyébként hibát jelez és újra bekéri a bemenetet.

Mátrix és egyéb kiíratő függvények

- ***void matrix_kiiratasa_konzol(Matrix* matrix) és void matrix_kiiratasa_file(Matrix* matrix, const char* filename) függvények:***

A két függvény működési elve azonos, a különbség az elnevezésből is adott, míg az egyik a standard kimenetre (konzolra) írja ki a mátrix érétkeit, addig a másik a paraméterként kapott nevű file-ba.

- **void double_eredmeny_kiiratasa_file(double eredmeny, const char* filename) és void int_eredmeny_kiiratasa_file(int eredmeny, const char* filename) függvények:**

Ebben az esetben is a két függvény gondolatmenete azonos, lényeges különbség, hogy az egyik egész típusú paramétert kap és azt írja a paraméterként kapott file-ba, a másik meg lebegőpontos értéket kap és azzal teszi meg ugyanezt.

- ***void cím() és void menu() függvények:***

Ezek a függvények a UI megjelenéséért felelősek. Az előbbi egy design elemnek szánt címet jelenít meg a standard kimeneten, az utóbbi pedig a menü elemeit.

A mátrix művelet függvények

Minden függvény egy, vagy két mátrixot kap paraméterként, elvégzi a szóbanforgó műveletet, majd visszatér az eredménnyel legyen az mátrix, egész, vagy valós típusú.
Kivétel az előbbi állítások alól az összes Gauss Eliminációs függvény, mert azok a paraméterként kapott ***Matrix*** típusú változók eredeti értékeit változtatják meg a meghívás helyén, és a ***Gauss_Megoldas*** három paraméterrel rendelkezik, melyből kettő tömb mert ott bejön a képletbe a szabadtagok tömbje és a megoldások tömbje is.

- ***Matrix* matrix_masolasa(Matrix* eredeti):***

  Lemásolja a kapott mátrixot egy segédváltozóba és visszatér vele.

- ***Matrix* matrixok_osszeadasa(Matrix* a, Matrix* b):***

  Kiszámolja a két mátrix összegét egy segédváltozóba és visszatér vele.

- ***Matrix* matrixok_kivonasa(Matrix* a, Matrix* b):***

  Kiszámolja a két mátrix különbségét egy segédváltozóba és visszatér vele.

- ***Matrix* matrixok_szorzasa(Matrix* a, Matrix* b):***

  Összeszoroz két mátrixot egy segédváltozóba és visszatér vele

- ***Matrix* matrix_transzponalasa(Matrix* a):***

  Kiszámítja egy mátrix transzponáltját egy segédváltozóba és visszatér vele.

- ***void Gauss_Eliminacio_det_rang(Matrix* a):***

Lefuttatja a Gauss Eliminációt egy mátrixra úgy, hogy a futás után a meghívás helyén a mátrix eredeti értéke, a mátrix felsőháromszöges alakjára változik, amelyből könnyen kiszámítható a determináns.

- ***void Gauss_Eliminacio_inverz(Matrix* a, Matrix* i):***

Lefuttatja a Gauss Eliminációt egy mátrixra úgy, hogy a futás után a meghívás helyén a mátrix eredeti értéke egy egységmátrix, és az inverz változó tartalmazni fogja a mátrix inverzét.

- ***Matrix* inverz_matrix(Matrix* a):***

Inicializál egy mátrixot amelyet egységmátrixra állít, majd meghívja a ***Gauss_Eliminacio_inverz*** függvényt, aminek segítségével az inicializált mátrixba bekerül a paraméterben szereplő mátrix inverze, majd visszatér ezzel a mátrixal.

- ***void Gauss_Megoldas(Matrix* a, double* b, double* x):***

Egy mátrixban kiszámítja az együttható mátrix inverzét, majd az inverz mátrixnak és a paraméterlistában érkező szabadtagok tömbjének segítségével kiszámolja egy segédtömbbel az ismeretleneket, amelyeket az x tömbbe ment és paraméteren keresztül téríti vissza.

- ***void sorok_kivonasa(Matrix* matrix, int cel_sor, int forras_sor, double factor):***

  Kivonja egy adott sorból a másik adott sor számszorosát egy mátrixban.

- ***void sorok_osztasa(Matrix* matrix, int sor, double oszto):***

  Leoszt egy adott sort egy adott valós számmal egy mátrixban.

- ***void sorok_csereje(Matrix* matrix, int sor1, int sor2):***

  Felcserél két sort egy mátrixban.

- ***double matrix_nyoma(Matrix* a):***

  Kiszámítja egy mátrix főátlóján található elemek összegét és visszatér vele.

- ***double determinans_szamitas(Matrix* a):***

Kiszámítja egy mátrix főátlóján található elemek szorzatát, de csak a **Gauss_Eliminacio_rang_det** függvény meghívása után. A függvény visszatér egy valós változóval, amibe a determináns lett kiszámolva.

- ***int matrix_rangja(Matrix* a);***

Megszámolja a nem nulla sorokat egy mátrixban, de csak a **Gauss_Eliminacio_rang_det** függvény meghívása után. A függvény visszatér egy egész típusú változóval, amibe a rang lett kiszámolva.

Menüpont függvények

- ***void file_vagy_billzet_1(Matrix **A);***

Megjeleníit egy újabb menüt, utána a beolvasott bemenet függvényében érvényesül egy **switch** utasítás megfelelő ága. Az egyik ágon file-ból beolvasó függvény kerül meghívásra és így a meghívás helyén lévő mátrix a file-beli méretek és értékek szerint jön létre, a másik ágon hasonlóan a felhasználó általi beolvasás függvényt hívja meg a program, és így a szóbanforgó mátrix értékei a felhasználó által megadott paraméterek szerint alakulnak.

- ***void file_vagy_billzet_2(Matrix **A, Matrix **B);***

Működése identikus az előző függvényével, a különbség annyi, hogy ez két mátrixra valósítja meg a leírtakat. Fontos, hogy külön ki lehet választani, hogy az első mátrix értékeit, hogy szeretnénk megadni és külön, hogy a második mátrix értékeit, hogy szeretnénk megadni.

Az előző függvények nem is valódi menüpont függvények, de elengedhetetlenek a menüpontok megvalósításában. A valódi menüpontokat egyenként értelmetlen lenne tárgyalni, mivel az eddigi függvények alkalmazásaiból állnak. Minden menüpontban szerepel a menüponthoz szükséges mátrixok és változók definiálása és a ***file_vagy_billzet_1*** vagy ***_2*** függvények meghívása (kivétel a ***haromszog_terulete_menupont*** és ***harom_pont_kollinearis_menupont*** ott a ***harom_pont_matrix*** függvény kerül meghívásra).

Tehát minden menüpont onnan indul, hogy későbbi függvényhívások számára szükséges paraméterek adottak és a mátrixok értékekkel vannak feltöltve. Továbbá menüpont funkciójától függően meghívásra kerül a megfelelő műveleti függvény és utána a kiíró függvények, hogy kiíródjanak az eredmények úgy file-ba, mint a standard kimenetre.

A megvalósított menüpont függvények:

- ***void szorzat_menupont();***
- ***void osszeg_menupont();***
- ***void kivonas_menupont();***
- ***void nyom_menupont();***
- ***void transzponalt_menupont();***
- ***void Gauss_Eliminacio_menupont();***
- ***void egyenletrendszer_megoldasa_menupont();***
- ***void determinans_menupont();***
- ***void haromszog_terulete_menupont();***
- ***void harom_pont_kollinearis_menupont();***
- ***void matrix_rangja_menupont();***
- ***void matrix_inverze_menupont();***

Főmenü függvény

A függvény meghívja a ***cím()*** függvényt, inicializál egy ***menupont*** nevű egész változót, amelyet később használ a felhasználó által választott menüpont tárolására. Létrehoz egy ***leall*** nevű logikai változót, amely alapértelmezetten igaz értéket kap. Egy végtelen ciklusban elérhetővé teszi a menüt és várja a felhasználó választását. A ***beolvasInt*** függvény segítségével beolvassa a felhasználó választott menüpontját. Egy ***switch*** utasítást használ a választott menüpont alapján, és hívja meg a megfelelő menüpont függvényt a kiválasztott művelet végrehajtásához.

A main függvény

A program betölti a fejlécfile-okat, hogy elérhetővé tegye a szükséges függvényeket és adattípusokat. Végül a ***fomenu*** függvényt hívja meg, ami a program fő kezelőmenüjét jeleníti meg.

A teljes függvénykönyvtár hibakezelése és memóriakezelése

Fontos megjegyezni, hogy a program összes függvénye, illetve komponense kezeli a lehetséges hibákat, legyen az memóriafoglalási, vagy hibás bemenet. Konzoli hibás bemenetek esetén figyelmezteti a felhasználót és újra bekéri a hibásan megadott értéket. File-ból beolvasás esetén pedig szintén jelzi, ha a file hibás bemeneteket tartalmaz és újra megjeleníti a főmenüt a felhasználónak.

A program gondosan odafigyel a memóriakezelésre is, mivel az dinamikusan valósul meg, bárhol szakadjon félbe a program mindig felszabadítja a lefoglalt memóriaterületeket, de persze abban az esetben sincs memóriaszivárgás, ha a program sikeresen fut.

Továbbá azt is fontos megjegyezni, hogy minden műveleti és menüpont függvényben ellenőrizve van, hogy a művelet a matematika szabályai szerint elvégezhetőek, ha nem akkor pedig erre is felhívja a felhasználó figyelmét.