A programozás alapjai 2.

Házi Feladat Dokumentáció

**Mátrix osztály**

Készítette: Pákozdi Bence ,BJAIKB

pakozdi.bence@mensa.hu

2023. 06. 02.

Tartalomjegyzék

[Felhasználói dokumentáció 3](#__RefHeading___Toc349_1544674224)

[Osztályok statikus leírása 4](#__RefHeading___Toc351_1544674224)

[Matrix osztály 4](#__RefHeading___Toc353_1544674224)

[Konstruktorok 4](#__RefHeading___Toc355_1544674224)

[Műveletek 4](#__RefHeading___Toc357_1544674224)

[Operátorok 4](#__RefHeading___Toc359_1544674224)

[Egyéb függvények 5](#__RefHeading___Toc361_1544674224)

[Vektor osztály 5](#__RefHeading___Toc363_1544674224)

[Konstruktorok 5](#__RefHeading___Toc365_1544674224)

[Műveletek 5](#__RefHeading___Toc367_1544674224)

[Operátorok 6](#__RefHeading___Toc369_1544674224)

[UML osztálydiagramm 7](#__RefHeading___Toc371_1544674224)

[Összegzés 8](#__RefHeading___Toc373_1544674224)

[Képernyőképek a futó alkalmazásról 9](#__RefHeading___Toc375_1544674224)

# Felhasználói dokumentáció

Az alkalmazás egy mátrix és vektor műveleteket támogató szoftver, amely lehetővé teszi a mátrixok és vektorok létrehozását, manipulálását és matematikai műveletek végrehajtását.

Az alkalmazás használatához először válasszuk ki a kívánt menüpontot az elérhető opciók közül:

1. **Mátrix kezelése**: Ebben a menüpontban lehetőségünk van új mátrixok létrehozására, meglévő mátrixok betöltésére fájlból vagy módosítására. Az itt elérhető műveletek a következők:
   * Új mátrix létrehozása: Megadhatjuk a mátrix sorainak és oszlopainak számát, majd megadhatjuk az egyes elemek értékeit.
   * Mátrix betöltése fájlból: Kiválaszthatunk egy fájlt, amely tartalmazza a mátrix adatait, és beolvassuk azt.
   * Mátrix módosítása: Megadhatjuk a már létező mátrixunk egyes elemének értékét.
2. **Mátrix műveletek**: Ebben a menüpontban különböző műveleteket hajthatunk végre a mátrixokon:
   * Mátrix összeadása: Kiválaszthatunk két mátrixot, majd összeadjuk őket.
   * Mátrix kivonása: Kiválaszthatunk két mátrixot, majd kivonjuk az egyiket a másiktól.
   * Mátrix szorzása: Kiválaszthatunk két mátrixot, majd elvégezzük a szorzást közöttük.
   * Mátrix determinánsa: Kiválaszthatunk egy mátrixot, majd kiszámítjuk annak determinánsát.
   * Mátrix transzponálása: Kiválaszthatunk egy mátrixot, majd elvégezzük a transzponálást.
3. **Mátrix kiírása fájlba**: Kiválaszthatunk egy mátrixot, majd kiírhatjuk annak tartalmát egy fájlba.
4. **Vektor kezelése**: Ebben a menüpontban lehetőségünk van új vektorok létrehozására, meglévő vektorok betöltésére fájlból vagy módosítására. Az itt elérhető műveletek a következők:
   * Új vektor létrehozása: Megadhatjuk a vektor elemeinek számát, majd megadhatjuk az egyes elemek értékeit.
   * Vektor betöltése fájlból: Kiválaszthatunk egy fájlt, amely tartalmazza a vektor adatait, és beolvassuk azt.
   * Vektor módosítása: Megadhatjuk a már létező vektorunk egyes elemének értékét.
5. **Vektor műveletek**: Ebben a menüpontban különböző műveleteket hajthatunk végre a vektorokon:
   * Vektor összeadása: Kiválaszthatunk két vektort, majd összeadjuk őket.
   * Vektor kivonása: Kiválaszthatunk két vektort, majd kivonjuk az egyiket a másiktól.
   * Vektor skalárral való szorzása: Kiválaszthatunk egy vektort és egy skalárt, majd elvégezzük a szorzást közöttük.
6. **Vektor kiírása fájlba**: Kiválaszthatunk egy vektort, majd kiírhatjuk annak tartalmát egy fájlba.

# Osztályok statikus leírása

## Matrix osztály

A Matrix osztály egy olyan osztály, amely egy kétdimenziós mátrixot reprezentál. Az osztály lehetővé teszi a mátrix alapvető műveleteinek végrehajtását, mint például a mátrix transzponálása, inverzének kiszámítása, determináns számítása, Gauss-Jordan elimináció és még sok más.

### Konstruktorok

* Matrix(): Alapértelmezett konstruktor, létrehoz egy üres mátrixot.
* Matrix(int n, int m): Konstruktor, létrehoz egy n sorból és m oszlopból álló mátrixot.
* Matrix(int n, int m, T\* arr, int arr\_size): Konstruktor, létrehoz egy n sorból és m oszlopból álló mátrixot, és feltölti a megadott tömbből.
* Matrix(const Matrix<T>& matrix): Másoló konstruktor, létrehoz egy új mátrixot a másolással.

### Műveletek

* T det() const: Kiszámítja és visszaadja a mátrix determinánsát.
* Matrix<T> inverse() const: Kiszámítja és visszaadja a mátrix inverzét.
* Matrix<T> getSubMatrix(int i, int j, int k, int l) const: Visszaadja a mátrix egy részét a megadott sor- és oszlopintervallumok alapján.
* void swapRows(int i, int j): Kicseréli a mátrix két sorát.
* void swapColumns(int i, int j): Kicseréli a mátrix két oszlopát.
* Matrix<T> mergeHorizontal(const Matrix<T>& matrix) const: Összefűzi a mátrixot egy másik mátrixszal vízszintesen.
* Matrix<T> mergeVertical(const Matrix<T>& matrix) const: Összefűzi a mátrixot egy másik mátrixszal függőlegesen.
* Matrix<T> transposition() const: Kiszámolja a mátrix transzponáltját.
* Matrix<T> GaussJordanElimination() const: Elvégzi a mátrixon a Gauss-Jordan eliminációt.
* void writeToFile(const std::string& filename): Kiírja a mátrix tartalmát egy fájlba.
* void readFromFile(const std::string& filename): Beolvassa a mátrix tartalmát egy fájlból.

### Operátorok

* Vektor<T> operator[](int i) const: Visszaadja a Matrix osztály i-edik sorát konstans hivatkozással.
* Vektor<T>& operator[](int i): Visszaadja a Matrix osztály i-edik sorát módosítható hivatkozással.
* Matrix<T> operator-(const Matrix<T>& matrix) const: Kivonja a megadott mátrixot a jelenlegi mátrixból és visszaadja az eredményt.
* Matrix<T> operator+(const Matrix<T>& matrix) const: Összeadja a megadott mátrixot a jelenlegi mátrixhoz és visszaadja az eredményt.
* Matrix<T> operator\*(const Matrix<T>& matrix) const: Megszorozza a jelenlegi mátrixot a megadott mátrixszal és visszaadja az eredményt.
* Matrix<T> operator\*(const T& scalar) const: Szorzást végez a jelenlegi mátrix elemeivel a megadott skalárral és visszaadja az eredményt.

### Egyéb függvények

* Matrix<T> unitMatrix(int size): Egy egységmátrixot hoz létre a megadott méret alapján.

## Vektor osztály

A Vektor osztály egy olyan osztály, amely egy egydimenziós vektort reprezentál. Az osztály tartalmazza a vektor elemeit és lehetővé teszi az elemek elérését és módosítását.

### Konstruktorok

* Vektor(): Alapértelmezett konstruktor, létrehoz egy üres vektort.
* Vektor(int n): Konstruktor, létrehoz egy n hosszú vektort.
* Vektor(int n, T\* arr, int arr\_size): Konstruktor, létrehoz egy n hosszú vektort, és feltölti a megadott tömbből.
* Vektor(const Vektor<T>& vector): Másoló konstruktor, létrehoz egy új vektort a másolással.

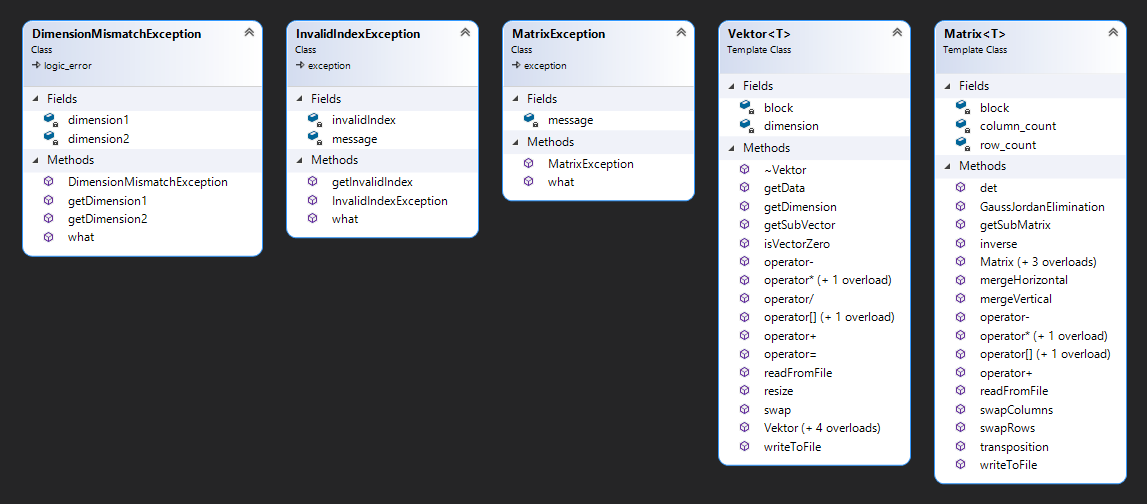
### Műveletek

* int size() const: Visszaadja a vektor méretét.
* T\* getData() const: Visszaadja a vektor elemeit tartalmazó tömbre mutató pointert.
* void resize(int n): Átméretezi a vektort a megadott méretre.
* void swap(int i, int j): Kicseréli a vektor i-edik és j-edik elemét.
* Vektor<T> getSubVector(int i, int j) const: Visszaadja a vektor egy részét a megadott kezdő- és végindexek alapján.
* T& operator[](int i): Visszaadja a Vektor osztály i-edik elemét módosítható hivatkozással.
* T operator[](int i) const: Visszaadja a Vektor osztály i-edik elemét konstans hivatkozással.

### Operátorok

* Vektor<T> operator-(const Vektor<T>& vector) const: Kivonja a megadott vektort a jelenlegi vektorból és visszaadja az eredményt.
* Vektor<T> operator+(const Vektor<T>& vector) const: Összeadja a megadott vektort a jelenlegi vektorhoz és visszaadja az eredményt.
* Vektor<T> operator\*(const T& scalar) const: Szorzást végez a vektor elemeivel a megadott skalárral és visszaadja az eredményt.

# UML osztálydiagramm



# Összegzés

**Tanulságok:**

A házi feladat elkészítése során többek között azt tanultam, hogy a sablon alapján történő osztály implementálása és az osztályok közötti kommunikáció hatékony módja lehet a programozásnak. Emellett a mátrix és vektor műveletek implementálása során az algoritmusok részletes megértése és helyes alkalmazása is fontos szerepet játszik.

**Nehézségek:**

Az implementáció során néhány nehézséggel szembesültem, például a mátrix inverzének számítása és a Gauss-Jordan elimináció algoritmusának megvalósítása kihívást jelentett. Azonban a részletes tervezés és a megfelelő dokumentáció segített átlendülni ezeken a nehézségeken.

**Továbbfejlesztési lehetőségek:**

Az alkalmazás továbbfejlesztése számos lehetőséget kínál. Például a felhasználói felület fejlesztése és interaktívabbá tétele segíthetne az alkalmazás használatának egyszerűsítésében. Emellett újabb matematikai műveletek és algoritmusok hozzáadása is bővíthetné az alkalmazás funkcionalitását. Az alkalmazás célokzönségét és alkalmazási területét is kibővíthetnénk, például a tudományos kutatásokban, pénzügyi elemzésekben vagy gépi tanulásban történő alkalmazásra is alkalmas lehetne.