readme.md

Postavka zadatka za implementaciju matrice

Cilj ovog zadatka je implementirati klasu Matrix koja predstavlja matricu elementa tipa T. Matrica se sastoji od redova i kolona, a elementi matrice se čuvaju u vektoru vektora. Klasa Matrix treba da implementira sljedeće metode:

Konstruktori

```
Matrix();
// konstruktor koji kreira praznu matricu (bez redova i kolona).
Matrix(size_t rows, size_t cols);
// konstruktor koji kreira matricu sa zadanim brojem redova i kolona.
// Svi elementi matrice se inicijaliziraju na podrazumijevanu vrijednost t
Matrix(const Matrix& other);
// copy konstruktor koji kopira sve elemente matrice other u novu matricu.
Matrix(Matrix&& other);
Move konstruktor koji preuzima sve elemente matrice other u novu matricu.
```

Operatori dodjeljivanja vrijednosti

```
Matrix& operator=(const Matrix& other);
// copy operator= koji kopira sve elemente matrice other u trenutnu matric
Matrix& operator=(Matrix&& other);
// move operator= koji preuzima sve elemente matrice other u trenutnu matr
```

Pristup elementima matrice

```
T& at(size_t row, size_t column);
// metoda koja vraća referencu na element matrice na poziciji (row, column
// Ukoliko su zadane koordinate van granica matrice, baca se iznimka tipa
std::vector<T>& operator[](size_t row);
// operator indeksiranja koji vraća referencu na red matrice sa indeksom r
```

Operatori poređenja

```
bool operator==(const Matrix& other) const;
// Logicki operator== koji poredi sve elemente dvije matrice i
// vraća true ako su svi elementi jednaki, u suprotnom vraća false.
bool operator!=(const Matrix& other) const;
// Logicki operator!= koji vraća negaciju operatora ==.
```

Operacije nad matricama

```
Matrix operator+(const Matrix& other) const;
// operator+ matrica koji vraća novu matricu koja je zbir dvije matrice.
Matrix operator*(const Matrix& other) const;
// operator* matrica koji vraća novu matricu koja je proizvod dvije matric
Matrix operator*(T factor) const;
// operator množenja skalarom koji vraća novu matricu čiji su svi elementi
```

Dijagonale matrice

```
std::vector<T> mainDiagonal() const;
// metoda koja vraća vektor sa elementima glavne dijagonale matrice.
std::vector<T> secondaryDiagonal() const;
// metoda koja vraća vektor sa elementima sporedne dijagonale matrice.
```

Informacije o matrici

```
std::pair<size_t, size_t> size() const;
// metoda koja vraća dimenzije matrice u obliku para (rows, columns).
```

Manipulacija matricom

```
Matrix& push_back(const std::vector<T>& row);
// metoda koja dodaje novi red u matricu sa elementima iz vektora row.
Matrix& push_back(const std::initializer_list<T>& row);
// metoda koja dodaje novi red u matricu sa elementima iz liste inicijaliz
Matrix& expand(const std::initializer_list<T>& row);
// metoda koja dodaje jedan element na kraj svakog reda matrice iz liste i
Matrix& expand(const std::vector<T>& row);
// metoda koja dodaje jedan element na kraj svakog reda matrice iz vektora
Matrix& expand(U&& element);
```

```
// metoda koja dodaje jedan element na kraj svakog reda matrice.
// Element se zadaje kao univerzalna referenca U&&.
```

- U implementaciji se mogu koristiti sljedeće strukture iz standardne biblioteke:

1. std::vector
2. std::pair
3. std::initializer_list

Klasa Matrix treba biti generička, to jest, tip elementa matrice T treba biti parametar klase.

Za čuvanje elemenata matrice koristi se std::vector<std::vector>. Ukoliko želite da optimizujete performanse, možete razmotriti i korištenje std::vector za čuvanje elemenata matrice.