GitHub Copilot

Generated by Doxygen 1.12.0

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 matrix Class Reference

Klasa reprezentuj¹ca kwadratow¹ macierz o dynamicznie alokowanym rozmiarze.

`#include <matrix.h>`

Public Member Functions

- matrix ()

  Konstruktor domylny. Tworzy pust¹ macierz.
- matrix (int n)

  Konstruktor tworz¹cy macierz o zadanym rozmiarze.
- matrix (int n, int ∗t)

  Konstruktor inicjalizuj¹cy macierz danymi z tablicy.
- matrix (const matrix &m)

  Konstruktor kopiuj¹cy. Tworzy now¹ macierz na podstawie istniej¹cej.
- ∼matrix ()

  Destruktor. Usuwa dynamicznie alokowan¹ pamiêæ.
- matrix & alokuj (int n)

  Alokuje pamiêæ dla macierzy o zadanym rozmiarze.
- matrix & wstaw (int x, int y, int wartosc)

  Wstawia wartoæ do okrelonej komórki macierzy.
- int pokaz (int x, int y)

  Pobiera wartoæ z okrelonej komórki macierzy.
- matrix & dowroc ()

  Transponuje macierz. Zamienia wiersze z kolumnami.
- matrix & losuj ()

  Wype^{3}nia macierz losowymi wartociami.
- matrix & losuj (int x)

  Wype^{3}nia czêæ macierzy losowymi wartociami.
- matrix & diagonalna (int ∗t)

  Tworzy macierz diagonaln¹ na podstawie tablicy wartoci.
- matrix & diagonalna_k (int k, int ∗t)

  Tworzy macierz z przesuniêt¹ przek¹tn¹.

- matrix & operator+ (matrix &m)

  Operatory matematyczne.
- matrix & operator∗ (matrix &m)

  Mno¿y dwie macierze.
- matrix & operator+ (int a)

  Dodaje liczbê do wszystkich elementów macierzy.
- matrix & operator∗ (int a)

  Mno¿y wszystkie elementy macierzy przez liczbê.
- bool operator== (const matrix &m) const

  Porównuje, czy dwie macierze s¹ równe.
- bool operator> (const matrix &m) const

  Porównuje, czy macierz jest wiêksza od innej.
- bool operator< (const matrix &m) const

  Porównuje, czy macierz jest mniejsza od innej.
- **matrix** (int n)
- **matrix** (int n, int ∗t)
- **matrix** (const matrix &m)
- matrix & **alokuj** (int n)
- matrix & **wstaw** (int x, int y, int wartosc)
- int **pokaz** (int x, int y)
- matrix & **dowroc** ()
- matrix & **losuj** ()
- matrix & **losuj** (int x)
- matrix & **diagonalna** (int ∗t)
- matrix & **diagonalna_k** (int k, int ∗t)
- matrix & **operator+** (matrix &m)
- matrix & **operator∗** (matrix &m)
- matrix & **operator+** (int a)
- matrix & **operator∗** (int a)
- bool **operator==** (const matrix &m) const
- bool **operator>** (const matrix &m) const
- bool **operator<** (const matrix &m) const

Friends

- std::ostream & operator<< (std::ostream &os, const matrix &m)

  Wypisuje zawartoæ macierzy na strumieñ wyjciowy.
- std::ostream & operator<< (std::ostream &os, const matrix &m)

  Operatory.

3.1.1 Detailed Description

Klasa reprezentuj¹ca kwadratow¹ macierz o dynamicznie alokowanym rozmiarze.

Klasa `matrix` umo¿liwia:

- Dynamiczne zarz¹dzanie pamiêci¹ dla macierzy.
- Manipulacjê wartociami w macierzy.
- Operacje matematyczne i logiczne.
- Tworzenie specjalnych uk^{3}adów, takich jak macierze diagonalne.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 matrix() [1/4]

```
matrix::matrix ()
```

Konstruktor domylny. Tworzy pust¹ macierz.

Implementacja konstruktorów.

Konstruktor domylny. Tworzy pust¹ macierz bez alokacji pamiêci.

3.1.2.2 matrix() [2/4]

```
matrix::matrix (
            int n)
```

Konstruktor tworz¹cy macierz o zadanym rozmiarze.

Parameters

n	Rozmiar macierzy (n x n).

3.1.2.3 matrix() [3/4]

```
matrix::matrix (
            int n,
            int * t)
```

Konstruktor inicjalizuj¹cy macierz danymi z tablicy.

Konstruktor inicjalizuj¹cy macierz danymi z tabeli.

Parameters

n	Rozmiar macierzy (n x n).
t	Tablica jednowymiarowa przechowuj¹ca dane macierzy.

3.1.2.4 matrix() [4/4]

```
matrix::matrix (
            const matrix & m)
```

Konstruktor kopiuj¹cy. Tworzy now¹ macierz na podstawie istniej¹cej.

Konstruktor kopiuj¹cy. Tworzy now¹ macierz jako kopiê istniej¹cej.

Parameters

m	Macierz do skopiowania.

3.1.2.5 ∼matrix()

```
matrix::∼matrix ()
```

Destruktor. Usuwa dynamicznie alokowan¹ pamiêæ.

Destruktor. Zwalnia dynamicznie alokowan¹ pamiêæ.

3.1.3 Member Function Documentation

3.1.3.1 alokuj()

```
matrix & matrix::alokuj (
            int size)
```

Alokuje pamiêæ dla macierzy o zadanym rozmiarze.

Operacje na macierzy.

Parameters

n	Rozmiar macierzy (n x n).

Returns

Referencja do obiektu `matrix`.

Alokuje pamiêæ dla macierzy o zadanym rozmiarze. Jeli pamiêæ jest ju¿ zaalokowana, sprawdza rozmiar i alokuje ponownie, jeli to konieczne.

Parameters

size	Rozmiar macierzy (n x n).

Returns

Referencja do obiektu `matrix`.

3.1.3.2 diagonalna()

```
matrix & matrix::diagonalna (
            int * t)
```

Tworzy macierz diagonaln¹ na podstawie tablicy wartoci.

Parameters

t	Tablica wartoci do wstawienia na przek¹tn¹.

Returns

Referencja do obiektu `matrix`.

3.1.3.3 diagonalna_k()

```
matrix & matrix::diagonalna_k (
            int k,
            int * t)
```

Tworzy macierz z przesuniêt¹ przek¹tn¹.

Parameters

k	Liczba okrelaj¹ca przesuniêcie (ujemna, zero, dodatnia).
t	Tablica wartoci do wstawienia na przek¹tn¹.

Returns

Referencja do obiektu `matrix`.

3.1.3.4 dowroc()

```
matrix & matrix::dowroc ()
```

Transponuje macierz. Zamienia wiersze z kolumnami.

Returns

Referencja do obiektu `matrix`.

3.1.3.5 losuj() [1/2]

```
matrix & matrix::losuj ()
```

Wype^{3}nia macierz losowymi wartociami.

Wype^{3}nia macierz losowymi wartociami z zakresu 0-9.

Returns

Referencja do obiektu `matrix`.

3.1.3.6 losuj() [2/2]

```
matrix & matrix::losuj (
            int x)
```

Wype^{3}nia czêæ macierzy losowymi wartociami.

Parameters

x	Liczba losowych wartoci do wstawienia.

Returns

Referencja do obiektu `matrix`.

3.1.3.7 operator∗() [1/2]

```
matrix & matrix::operator* (
            int a)
```

Mno¿y wszystkie elementy macierzy przez liczbê.

Parameters

a	Liczba do mno¿enia.

Returns

Wynik operacji jako nowy obiekt `matrix`.

3.1.3.8 operator∗() [2/2]

```
matrix & matrix::operator* (
            matrix & m)
```

Mno¿y dwie macierze.

Parameters

m	Druga macierz.

Returns

Wynik mno¿enia jako nowy obiekt `matrix`.

Exceptions

std::invalid_argument	Jeli rozmiary macierzy s¹ niezgodne.

3.1.3.9 operator+() [1/2]

```
matrix & matrix::operator+ (
            int a)
```

Dodaje liczbê do wszystkich elementów macierzy.

Parameters

a	Liczba do dodania.

Returns

Wynik operacji jako nowy obiekt `matrix`.

3.1.3.10 operator+() [2/2]

```
matrix & matrix::operator+ (
            matrix & m)
```

Operatory matematyczne.

Dodaje dwie macierze.

Parameters

m	Druga macierz.

Returns

Wynik dodawania jako nowy obiekt `matrix`.

Exceptions

std::invalid_argument	Jeli rozmiary macierzy s¹ ró¿ne.

3.1.3.11 operator<()

```
bool matrix::operator< (
            const matrix & m) const
```

Porównuje, czy macierz jest mniejsza od innej.

Parameters

m	Druga macierz.

Returns

`true` jeli macierz jest mniejsza, `false` w przeciwnym razie.

3.1.3.12 operator==()

```
bool matrix::operator== (
            const matrix & m) const
```

Porównuje, czy dwie macierze s¹ równe.

Parameters

m	Druga macierz.

Returns

`true` jeli macierze s¹ równe, `false` w przeciwnym razie.

3.1.3.13 operator>()

```
bool matrix::operator> (
            const matrix & m) const
```

Porównuje, czy macierz jest wiêksza od innej.

Parameters

m	Druga macierz.

Returns

`true` jeli macierz jest wiêksza, `false` w przeciwnym razie.

3.1.3.14 pokaz()

```
int matrix::pokaz (
            int x,
            int y)
```

Pobiera wartoæ z okrelonej komórki macierzy.

Parameters

x	Indeks wiersza.
y	Indeks kolumny.

Returns

Wartoæ z komórki (x, y).

Exceptions

std::out_of_range	Jeli indeksy s¹ poza zakresem.

Parameters

x	Indeks wiersza.
y	Indeks kolumny.

Returns

Wartoæ w komórce (x, y).

Exceptions

std::out_of_range	Jeli podano indeks poza zakresem.

3.1.3.15 wstaw()

```
matrix & matrix::wstaw (
            int x,
            int y,
            int wartosc)
```

Wstawia wartoæ do okrelonej komórki macierzy.

Parameters

x	Indeks wiersza.
y	Indeks kolumny.
wartosc	Wartoæ do wstawienia.

Returns

Referencja do obiektu `matrix`.

Exceptions

std::out_of_range	Jeli indeksy s¹ poza zakresem.

Parameters

x	Indeks wiersza.
y	Indeks kolumny.
wartosc	Wartoæ do wstawienia.

Returns

Referencja do obiektu `matrix`.

Exceptions

std::out_of_range	Jeli podano indeks poza zakresem.

3.1.4 Friends And Related Symbol Documentation

3.1.4.1 operator<< [1/2]

```
std::ostream & operator<< (
            std::ostream & os,
            const matrix & m)  [friend]
```

Wypisuje zawartoæ macierzy na strumieñ wyjciowy.

Parameters

os	Strumieñ wyjciowy.
m	Macierz do wypisania.

Returns

Referencja do strumienia wyjciowego.

Operator wypisywania macierzy do strumienia.

Parameters

os	Strumieñ wyjciowy.
m	Macierz do wypisania.

Returns

Referencja do strumienia wyjciowego.

3.1.4.2 operator<< [2/2]

```
std::ostream & operator<< (
            std::ostream & os,
            const matrix & m)  [friend]
```

Operatory.

Wypisuje zawartoæ macierzy na strumieñ wyjciowy.

Operator wypisywania macierzy do strumienia.

Parameters

os	Strumieñ wyjciowy.
m	Macierz do wypisania.

Returns

Referencja do strumienia wyjciowego.

The documentation for this class was generated from the following files:

- Zadanie4/matrix.h
- Zadanie4/NagÅÃ^{3}wek1.h
- Zadanie4/matrix.cpp

Chapter 4

File Documentation

4.1 Zadanie4/main.cpp File Reference

Główny plik programu testującego klasę `matrix`.

```
#include <iostream>
#include "matrix.h"
```

Functions

- int main ()

  Funkcja główna programu.

4.1.1 Detailed Description

Główny plik programu testującego klasę `matrix`.

Plik zawiera funkcję `main()`, która testuje działanie klasy `matrix`. Tworzone są różne przypadki użycia, w tym:

- Tworzenie macierzy na podstawie tabeli wartości.
- Transpozycja macierzy za pomocą metody `dowroc()`.
- Wypełnianie macierzy losowymi wartościami za pomocą metody `losuj()`.

Dokumentacja zawiera przykłady użycia, generuje wykres wywołań oraz tabelę wyników.

4.1.2 Function Documentation

4.1.2.1 main()

```
int main ()
```

Funkcja główna programu.

Funkcja tworzy obiekt klasy `matrix`, a następnie wykonuje następujące operacje:

- Inicjalizacja macierzy o rozmiarze 4x4 z danymi z tablicy `table`.
- Wypisanie początkowej macierzy.
- Transpozycja macierzy za pomocą metody `dowroc()` i jej ponowne wypisanie.
- Losowe wypełnienie macierzy za pomocą metody `losuj()` i wypisanie wyników.

Returns

Zwraca `0`, jeśli program zakończył się poprawnie.

$<$ Rozmiar macierzy (4x4).

Tworzenie obiektu macierzy na podstawie tablicy danych.

Transpozycja macierzy.

Wypełnienie macierzy losowymi wartościami.

4.2 Zadanie4/matrix.cpp File Reference

Implementacja metod klasy `matrix`.

```
#include "matrix.h"
```

Functions

- std::ostream & operator$<<$ (std::ostream &os, const matrix &m)
  Operatory.

4.2.1 Detailed Description

Implementacja metod klasy `matrix`.

Plik zawiera definicje funkcji dla klasy `matrix`, takich jak:

- Dynamiczna alokacja i zwalnianie pamiêci (`allocateMemory()`, `freeMemory()`).
- Konstruktor i destruktor klasy.
- Operacje na macierzy: wstawianie, pobieranie, transpozycja, losowanie wartoci.
- Przeci[1]¿ony operator do wypisywania macierzy.

4.2.2 Function Documentation

4.2.2.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const matrix & m)
```

Operatory.

Wypisuje zawartoæ macierzy na strumieñ wyjciowy.

Operator wypisywania macierzy do strumienia.

Parameters

os	Strumieñ wyjciowy.
m	Macierz do wypisania.

Returns

Referencja do strumienia wyjciowego.

4.3 Zadanie4/matrix.h File Reference

Deklaracja klasy `matrix`.

```
#include <iostream>
#include <vector>
#include <stdexcept>
#include <cstdlib>
#include <ctime>
```

Classes

- class matrix

  Klasa reprezentuj¹ca kwadratow¹ macierz o dynamicznie alokowanym rozmiarze.

4.3.1 Detailed Description

Deklaracja klasy `matrix`.

Plik nag^{3}ówkowy zawiera deklaracjê klasy `matrix`, która umo¿liwia dynamiczne zarz¹dzanie kwadratow¹ macierz¹ o wymiarach `n` x `n`. Klasa zawiera metody do manipulacji danymi, takie jak wstawianie, pobieranie wartoci, transpozycja, czy losowe wype^{3}nianie. W pliku znajduj¹ siê równie¿ przeci¹¿one operatory do pracy z macierzami.

4.4 matrix.h

Go to the documentation of this file.

```
00001 #ifndef MATRIX_H
00002 #define MATRIX_H
00003
00004 #include <iostream>
00005 #include <vector>
00006 #include <stdexcept>
00007 #include <cstdlib>
00008 #include <ctime>
00009
00030 class matrix {
00031 private:
00032     int n;
00033     int** data;
00034
00038     void freeMemory();
00039
00044     void allocateMemory(int size);
00045
00046 public:
00051     matrix();
00052
00057     matrix(int n);
00058
00064     matrix(int n, int* t);
00065
00071     matrix(const matrix& m);
00072
00077     ~matrix();
00078
00084     matrix& alokuj(int n);
00085
00094     matrix& wstaw(int x, int y, int wartosc);
00095
00103     int pokaz(int x, int y);
00104
00110     matrix& dowroc();
00111
00116     matrix& losuj();
00117
00123     matrix& losuj(int x);
00124
00130     matrix& diagonalna(int* t);
00131
00138     matrix& diagonalna_k(int k, int* t);
00139
00141
00148     matrix& operator+(matrix& m);
00149
00156     matrix& operator*(matrix& m);
00157
00163     matrix& operator+(int a);
00164
00170     matrix& operator*(int a);
00171
00177     bool operator==(const matrix& m) const;
00178
00184     bool operator>(const matrix& m) const;
00185
00191     bool operator<(const matrix& m) const;
00192
00199     friend std::ostream& operator«(std::ostream& os, const matrix& m);
00200 };
00201
00202 #endif
```

4.5 NagÅÃ^{3}wek1.h

```
00001 #ifndef MATRIX_H
00002 #define MATRIX_H
00003
00004 #include <iostream>
00005 #include <vector>
00006 #include <stdexcept>
00007 #include <cstdlib>
00008 #include <ctime>
00009
00010 class matrix {
00011 private:
```

```
00012     int n;          // Rozmiar macierzy n x n
00013     int** data;     // Dynamiczna alokacja danych macierzy
00014
00015     void freeMemory();  // Pomocnicza metoda do zwalniania pamiêci
00016     void allocateMemory(int size); // Pomocnicza metoda do alokacji pamiêci
00017
00018 public:
00019     matrix();                       // Konstruktor domylny
00020     matrix(int n);                  // Konstruktor o wymiarach n x n
00021     matrix(int n, int* t);          // Konstruktor z tabel¹ danych
00022     matrix(const matrix& m);        // Konstruktor kopiuj¹cy
00023     ~matrix();                      // Destruktor
00024
00025     matrix& alokuj(int n);          // Alokacja pamiêci
00026     matrix& wstaw(int x, int y, int wartosc);  // Wstawianie wartoci
00027     int pokaz(int x, int y);        // Pobieranie wartoci
00028     matrix& dowroc();               // Transponowanie macierzy
00029     matrix& losuj();                // Losowe wype³nienie macierzy
00030     matrix& losuj(int x);           // Losowe wype³nienie czêci macierzy
00031     matrix& diagonalna(int* t);     // Macierz diagonalna
00032     matrix& diagonalna_k(int k, int* t); // Przek¹tna przesuniêta
00033
00034     // Operatory
00035     matrix& operator+(matrix& m);  // Dodawanie macierzy
00036     matrix& operator*(matrix& m);  // Mno¿enie macierzy
00037     matrix& operator+(int a);      // Dodawanie liczby
00038     matrix& operator*(int a);      // Mno¿enie przez liczbê
00039     bool operator==(const matrix& m) const; // Porównanie równoci
00040     bool operator>(const matrix& m) const;  // Porównanie wielkoci
00041     bool operator<(const matrix& m) const;  // Porównanie mniejszoci
00042
00043     friend std::ostream& operator«(std::ostream& os, const matrix& m); // Wypisanie macierzy
00044 };
00045
00046 #endif
```

Index