proj4

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 files with brief descriptions:

Chapter 3

Class Documentation

3.1 Matrix< T > Class Template Reference

Klasa reprezentująca macierz.

`#include <Matrix.hpp>`

Public Member Functions

- Matrix (void)
  Konstruktor domyślny.
- Matrix (int n)
  Konstruktor tworzący macierz o rozmiarze n x n.
- Matrix (int n, int *t)
  Konstruktor tworzący macierz o rozmiarze n x n i wypełniający ją danymi z tablicy.
- Matrix (Matrix const &m)
  Konstruktor kopiujący.
- void wypisz (void)
  Wypisuje zawartość macierzy.
- Matrix & alokuj (int n)
  Alokuje pamięć dla macierzy o rozmiarze n x n.
- int pokaz (int x, int y) const
  Zwraca wartość elementu macierzy na pozycji (x, y).
- Matrix & dowroc (void)
  Odwraca macierz (transpozycja).
- Matrix & losuj (void)
  Wypełnia macierz losowymi wartościami od 0 do 9.
- Matrix & losuj (int x)
  Wypełnia x losowych elementów macierzy wartościami od 0 do 9.
- Matrix & diagonalna (const int *t)
  Wypełnia przekątną macierzy wartościami z tablicy t, pozostałe elementy są równe 0.
- Matrix & diagonalna_k (int k, const int *t)
  Wypełnia przekątną macierzy wartościami z tablicy t, przesuniętą o k pozycji.
- Matrix & kolumna (int x, const int *t)
  Wypełnia kolumnę x wartościami z tablicy t.

- Matrix & wiersz (int x, const int *t)

  Wypełnia wiersz x wartościami z tablicy t.
- Matrix & przekatna (void)

  Wypełnia macierz: 1 na przekątnej, 0 poza przekątną.
- Matrix & pod_przekatna (void)

  Wypełnia macierz: 1 pod przekątną, 0 nad przekątną i na przekątnej.
- Matrix & nad_przekatna (void)

  Wypełnia macierz: 1 nad przekątną, 0 pod przekątną i na przekątnej.

3.1.1 Detailed Description

template<typename T>
class Matrix< T >

Klasa reprezentująca macierz.

Template Parameters

T	Typ danych przechowywanych w macierzy.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Matrix() [1/4]

```
template<typename T >
Matrix< T >::Matrix (
            void )  [inline]
```

Konstruktor domyślny.

3.1.2.2 Matrix() [2/4]

```
template<typename T >
Matrix< T >::Matrix (
            int n)  [inline]
```

Konstruktor tworzący macierz o rozmiarze n x n.

Parameters

n	Rozmiar macierzy.

3.1.2.3 Matrix() [3/4]

```
template<typename T >
Matrix< T >::Matrix (
            int n,
            int * t)  [inline]
```

Konstruktor tworzący macierz o rozmiarze n x n i wypełniający ją danymi z tablicy.

Parameters

n	Rozmiar macierzy.
t	Wskaźnik do tablicy z danymi.

3.1.2.4 Matrix() [4/4]

```
template<typename T >
Matrix< T >::Matrix (
            Matrix< T > const & m)  [inline]
```

Konstruktor kopiujący.

Parameters

m	Obiekt macierzy do skopiowania.

3.1.3 Member Function Documentation

3.1.3.1 alokuj()

```
template<typename T >
Matrix & Matrix< T >::alokuj (
            int n)  [inline]
```

Alokuje pamięć dla macierzy o rozmiarze n x n.

Parameters

n	Rozmiar macierzy.

Returns

Referencja do obiektu macierzy.

3.1.3.2 diagonalna()

```
template<typename T >
Matrix & Matrix< T >::diagonalna (
            const int * t)  [inline]
```

Wypełnia przekątną macierzy wartościami z tablicy t, pozostałe elementy są równe 0.

Parameters

t	Wskaźnik do tablicy z wartościami.

Returns

Referencja do obiektu macierzy.

3.1.3.3 diagonalna_k()

```
template<typename T >
Matrix & Matrix< T >::diagonalna_k (
            int k,
            const int * t)  [inline]
```

Wypełnia przekątną macierzy wartościami z tablicy t, przesuniętą o k pozycji.

Parameters

k	Przesunięcie przekątnej.
t	Wskaźnik do tablicy z wartościami.

Returns

Referencja do obiektu macierzy.

3.1.3.4 dowroc()

```
template<typename T >
Matrix & Matrix< T >::dowroc (
            void )  [inline]
```

Odwraca macierz (transpozycja).

Returns

Referencja do obiektu macierzy.

3.1.3.5 kolumna()

```
template<typename T >
Matrix & Matrix< T >::kolumna (
            int x,
            const int * t)  [inline]
```

Wypełnia kolumnę x wartościami z tablicy t.

Parameters

x	Numer kolumny.
t	Wskaźnik do tablicy z wartościami.

Returns

Referencja do obiektu macierzy.

3.1.3.6 losuj() [1/2]

```
template<typename T >
Matrix & Matrix< T >::losuj (
            int x)  [inline]
```

Wypełnia x losowych elementów macierzy wartościami od 0 do 9.

Parameters

x	Liczba elementów do wypełnienia.

Returns

Referencja do obiektu macierzy.

3.1.3.7 losuj() [2/2]

```
template<typename T >
Matrix & Matrix< T >::losuj (
            void )  [inline]
```

Wypełnia macierz losowymi wartościami od 0 do 9.

Returns

Referencja do obiektu macierzy.

3.1.3.8 nad_przekatna()

```
template<typename T >
Matrix & Matrix< T >::nad_przekatna (
            void )  [inline]
```

Wypełnia macierz: 1 nad przekątną, 0 pod przekątną i na przekątnej.

Returns

Referencja do obiektu macierzy.

3.1.3.9 pod_przekatna()

```
template<typename T >
Matrix & Matrix< T >::pod_przekatna (
            void )  [inline]
```

Wypełnia macierz: 1 pod przekątną, 0 nad przekątną i na przekątnej.

Returns

Referencja do obiektu macierzy.

3.1.3.10 pokaz()

```
template<typename T >
int Matrix< T >::pokaz (
            int x,
            int y) const  [inline]
```

Zwraca wartość elementu macierzy na pozycji (x, y).

Parameters

x	Wiersz.
y	Kolumna.

Returns

Wartość elementu macierzy.

3.1.3.11 przekatna()

```
template<typename T >
Matrix & Matrix< T >::przekatna (
            void )  [inline]
```

Wypełnia macierz: 1 na przekątnej, 0 poza przekątną.

Returns

Referencja do obiektu macierzy.

3.1.3.12 wiersz()

```
template<typename T >
Matrix & Matrix< T >::wiersz (
            int x,
            const int * t)  [inline]
```

Wypełnia wiersz x wartościami z tablicy t.

Parameters

x	Numer wiersza.
t	Wskaźnik do tablicy z wartościami.

Returns

Referencja do obiektu macierzy.

3.1.3.13 wypisz()

```
template<typename T >
void Matrix< T >::wypisz (
            void )  [inline]
```

Wypisuje zawartość macierzy.

Chapter 4

File Documentation

4.1 src/main.cpp File Reference

```
#include "Matrix.hpp"
```

Functions

- int main (int argc, char ∗argv[ ])

4.1.1 Function Documentation

4.1.1.1 main()

```
int main (
            int argc,
            char * argv[ ])
```

4.2 src/Matrix.hpp File Reference

```
#include <random>
#include <vector>
#include <print>
```

Classes

- class Matrix< T >

  Klasa reprezentująca macierz.

4.3 Matrix.hpp

Go to the documentation of this file.

```
00001 #include <random>
00002 #include <vector>
00003 #include <print>
00004
00010 template <typename T>
00011 class Matrix {
00012 private:
00013     std::vector<std::vector<T» data;
00014
00015 public:
00019     Matrix(void) {}
00020
00026     Matrix(int n) : data(n, std::vector<T>(n)) {}
00027
00034     Matrix(int n, int* t) : data(n, std::vector<T>(n)) {
00035         for(auto& row : data){
00036             row = {t, t + n};
00037             t += n;
00038         }
00039     }
00040
00046     Matrix(Matrix const& m) : data(m.data) {}
00047
00051     void wypisz(void) {
00052         for(const auto& row : data){
00053             for(const auto& elem : row) {
00054                 std::print("{} ", elem);
00055             }
00056             std::println();
00057         }
00058     }
00059
00066     Matrix& alokuj(int n){
00067         if(data.size() == 0){
00068             data.resize(n, std::vector<T>(n));
00069         } else {
00070             if(data.size() < n){
00071                 data.resize(n, std::vector<T>(n));
00072             }
00073         }
00074         return *this;
00075     }
00076
00084     int pokaz(int x, int y) const {
00085         return data[x][y];
00086     }
00087
00093     Matrix& dowroc(void) {
00094         Matrix<T> temp(data.size());
00095
00096         for(int i = 0; i < data.size(); i++){
00097             for(int j = 0; j < data.size(); j++){
00098                 temp.data[i][j] = data[j][i];
00099             }
00100         }
00101
00102         *this = temp;
00103         return *this;
00104     }
00105
00111     Matrix& losuj(void) {
00112         std::random_device rd;
00113         std::mt19937 gen(rd());
00114         std::uniform_int_distribution dis(0, 9);
00115
00116         for(int i = 0; i < data.size(); i++){
00117             for(int j = 0; j < data.size(); j++){
00118                 data[i][j] = dis(gen);
00119             }
00120         }
00121         return *this;
00122     }
00123
00130     Matrix& losuj(int x) {
00131         std::random_device rd;
00132         std::mt19937 gen(rd());
00133         std::uniform_int_distribution<> dis(0, 9);
00134
00135         for(int i = 0; i < x; i++){
00136             int a = dis(gen);
00137             int b = dis(gen);
00138             data[a][b] = dis(gen);
```

```
00139         }
00140         return *this;
00141     }
00142
00149     Matrix& diagonalna(const int* t) {
00150         for(int i = 0; i < data.size(); i++){
00151             for(int j = 0; j < data.size(); j++){
00152                 if(i == j){
00153                     data[i][j] = t[i];
00154                 } else {
00155                     data[i][j] = 0;
00156                 }
00157             }
00158         }
00159         return *this;
00160     }
00161
00169     Matrix& diagonalna_k(int k, const int* t) {
00170         for(int i = 0; i < data.size(); i++){
00171             for(int j = 0; j < data.size(); j++){
00172                 if(i == j + k){
00173                     data[i][j] = t[j];
00174                 } else {
00175                     data[i][j] = 0;
00176                 }
00177             }
00178         }
00179         return *this;
00180     }
00181
00189     Matrix& kolumna(int x, const int* t) {
00190         for(int i = 0; i < data.size(); i++){
00191             data[i][x] = t[i];
00192         }
00193         return *this;
00194     }
00195
00203     Matrix& wiersz(int x, const int* t) {
00204         for(int i = 0; i < data.size(); i++){
00205             data[x][i] = t[i];
00206         }
00207         return *this;
00208     }
00209
00215     Matrix& przekatna(void) {
00216         for(int i = 0; i < data.size(); i++){
00217             for(int j = 0; j < data.size(); j++){
00218                 if(i == j){
00219                     data[i][j] = 1;
00220                 } else {
00221                     data[i][j] = 0;
00222                 }
00223             }
00224         }
00225         return *this;
00226     }
00227
00233     Matrix& pod_przekatna(void) {
00234         for(int i = 0; i < data.size(); i++){
00235             for(int j = 0; j < data.size(); j++){
00236                 if(i > j){
00237                     data[i][j] = 1;
00238                 } else {
00239                     data[i][j] = 0;
00240                 }
00241             }
00242         }
00243         return *this;
00244     }
00245
00251     Matrix& nad_przekatna(void) {
00252         for(int i = 0; i < data.size(); i++){
00253             for(int j = 0; j < data.size(); j++){
00254                 if(i < j){
00255                     data[i][j] = 1;
00256                 } else {
00257                     data[i][j] = 0;
00258                 }
00259             }
00260         }
00261         return *this;
00262     }
00263 };
```

Index