Matrix

1.0

Wygenerowano Wt, 13 sty 2026 09:27:20 dla Matrix za pomocą Doxygen 1.12.0

Wt, 13 sty 2026 09:27:20

Rozdział 1

Indeks klas

1.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 2

Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich plików wraz z ich krótkimi opisami:

Rozdział 3

Dokumentacja klas

3.1 Dokumentacja klasy matrix

Klasa reprezentująca kwadratową macierz liczb całkowitych.

`#include <matrix.h>`

Metody publiczne

- matrix (void)

  Konstruktor domyślny tworzący pustą macierz 0×0.
- matrix (int n)

  Konstruktor tworzący macierz n×n wypełnioną zerami.
- matrix (int n, int ∗t)

  Konstruktor tworzący macierz n×n z danymi z tablicy.
- matrix (const matrix &m)

  Konstruktor kopiujący - tworzy głęboką kopię macierzy.
- matrix (matrix &&other) noexcept=default

  Konstruktor przenoszący (domyślny)
- ∼matrix (void)

  Destruktor.
- int & at (int x, int y)

  Zwraca referencję do elementu macierzy z walidacją
- const int & at (int x, int y) const

  Zwraca stałą referencję do elementu macierzy.
- int pokaz (int x, int y) const

  Zwraca wartość elementu macierzy (alias dla at)
- matrix & alokuj (int n)

  Alokuje lub realokuje pamięć dla macierzy o nowym rozmiarze.
- matrix & odwroc (void)

  Transponuje macierz (zamienia wiersze z kolumnami)
- matrix & losuj (void)

  Wypełnia macierz losowymi liczbami z zakresu [0, 9].
- matrix & losuj (int x)

  Wypełnia macierz losowymi liczbami z zakresu [0, x].

- matrix & diagonalna (int ∗t)

  Ustawia wartości na głównej przekątnej macierzy.
- matrix & diagonalna_k (int k, int ∗t)

  Ustawia wartości na k-tej przekątnej macierzy.
- matrix & kolumna (int x, int ∗t)

  Ustawia wartości w wybranej kolumnie.
- matrix & wiersz (int y, int ∗t)

  Ustawia wartości w wybranym wierszu.
- matrix & przekatna (void)

  Tworzy macierz jednostkową (1 na przekątnej, 0 poza)
- matrix & pod_przekatna (void)

  Wypełnia jedynkami obszar pod główną przekątną
- matrix & nad_przekatna (void)

  Wypełnia jedynkami obszar nad główną przekątną
- matrix & szachownica (void)

  Wypełnia macierz wzorem szachownicy (naprzemienne 0 i 1)
- matrix & operator+= (int a)

  Dodaje skalar do wszystkich elementów macierzy.
- matrix & operator-= (int a)

  Odejmuje skalar od wszystkich elementów macierzy.
- matrix & operator∗= (int a)

  Mnoży wszystkie elementy macierzy przez skalar.
- matrix operator++ (int)

  Postinkrementacja - zwiększa wszystkie elementy o 1.
- matrix operator-- (int)

  Postdekrementacja - zmniejsza wszystkie elementy o 1.
- matrix & operator() (double d)

  Operator wywołania - dodaje wartość do wszystkich elementów.
- bool operator== (const matrix &m) const

  Sprawdza równość dwóch macierzy.
- bool operator> (const matrix &m) const

  Sprawdza czy wszystkie elementy są większe.
- bool operator< (const matrix &m) const

  Sprawdza czy wszystkie elementy są mniejsze.
- void wstaw (int x, int y, int val)

  Wstawia wartość do określonej pozycji w macierzy.
- int getSize () const

  Zwraca rozmiar macierzy.

Atrybuty prywatne

- int n

  Aktualny rozmiar macierzy (n×n)
- int allocated_n

  Rozmiar zaalokowanej pamięci.
- std::unique_ptr< int[ ]> macierz_ptr

  Wskaźnik do danych macierzy (przechowywane wierszami)

Przyjaciele

- matrix operator+ (const matrix &m1, const matrix &m2)

  Dodaje dwie macierze element po elemencie.
- matrix operator∗ (const matrix &m1, const matrix &m2)

  Wykonuje mnożenie macierzowe.
- matrix operator+ (const matrix &m, int a)

  Dodaje skalar do macierzy (macierz + liczba)
- matrix operator+ (int a, const matrix &m)

  Dodaje skalar do macierzy (liczba + macierz)
- matrix operator∗ (const matrix &m, int a)

  Mnoży macierz przez skalar (macierz ∗ liczba)
- matrix operator∗ (int a, const matrix &m)

  Mnoży skalar przez macierz (liczba ∗ macierz)
- matrix operator- (const matrix &m, int a)

  Odejmuje skalar od macierzy (macierz - liczba)
- matrix operator- (int a, const matrix &m)

  Odejmuje macierz od skalara (liczba - macierz)
- std::ostream & operator<< (std::ostream &o, const matrix &m)

  Operator wyjścia - wypisuje macierz do strumienia.

3.1.1 Opis szczegółowy

Klasa reprezentująca kwadratową macierz liczb całkowitych.

Klasa zapewnia pełną funkcjonalność operacji na macierzach kwadratowych, w tym operacje arytmetyczne, porównania, transformacje i wypełnianie wzorami. Pamięć jest zarządzana automatycznie przez std::unique_ptr.

Definicja w linii 23 pliku matrix.h.

3.1.2 Dokumentacja konstruktora i destruktora

3.1.2.1 matrix() [1/5]

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

Konstruktor domyślny tworzący pustą macierz 0×0.

Konstruktor domyślny - tworzy pustą macierz o rozmiarze 0x0.

Definicja w linii 15 pliku matrix.cpp.

3.1.2.2 matrix() [2/5]

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

Konstruktor tworzący macierz n×n wypełnioną zerami.

Konstruktor parametryczny - tworzy macierz n×n wypełnioną zerami.

Parametry

	n	Rozmiar macierzy kwadratowej

Definicja w linii 21 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

3.1.2.3 matrix() [3/5]

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

Konstruktor tworzący macierz n×n z danymi z tablicy.

Konstruktor z tablicą - tworzy macierz n×n wypełnioną wartościami z tablicy.

Parametry

	n	Rozmiar macierzy kwadratowej
	t	Wskaźnik do tablicy zawierającej n∗n elementów
	n	Rozmiar macierzy kwadratowej
	t	Wskaźnik do tablicy z danymi (wymaga n∗n elementów)

Definicja w linii 33 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

3.1.2.4 matrix() [4/5]

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

Konstruktor kopiujący - tworzy głęboką kopię macierzy.

Parametry

	m	Macierz źródłowa do skopiowania

Definicja w linii 44 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

3.1.2.5 matrix() [5/5]

```
matrix::matrix (
            matrix && other)  [default], [noexcept]
```

Konstruktor przenoszący (domyślny)

Parametry

	other	Macierz do przeniesienia

3.1.2.6 ~matrix()

```
matrix::~matrix (
            void )
```

Destruktor.

Destruktor - zwalnia automatycznie pamięć dzięki unique_ptr.

Definicja w linii 54 pliku matrix.cpp.

3.1.3 Dokumentacja funkcji składowych

3.1.3.1 alokuj()

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

Alokuje lub realokuje pamięć dla macierzy o nowym rozmiarze.

Parametry

	n	Nowy rozmiar macierzy (musi być $>$ 0)

Zwraca

Referencja do bieżącej macierzy (umożliwia łańcuchowanie)

Wyjątki

	std::logic_error	Jeśli rozmiar $<=$ 0

Jeżeli macierz nie ma jeszcze zaalokowanej pamięci lub aktualnie zaalokowana pamięć jest mniejsza niż wymagana, pamięć zostaje zaalokowana ponownie w rozmiarze rozmiar × rozmiar. Jeżeli zaalokowanej pamięci jest więcej niż potrzeba, istniejąca alokacja zostaje zachowana bez zmian.

Parametry

	rozmiar	Nowy rozmiar logiczny macierzy (musi być $>$ 0)

Zwraca

Referencja do bieżącego obiektu macierzy

Wyjątki

	std::logic_error	Gdy rozmiar <= 0

Definicja w linii 320 pliku matrix.cpp.

Odwołuje się do allocated_n, macierz_ptr i n.

Odwołania w main().

3.1.3.2 at() [1/2]

```
int & matrix::at (
            int x,
            int y)
```

Zwraca referencję do elementu macierzy z walidacją

Parametry

	x	Indeks wiersza (0-based)
	y	Indeks kolumny (0-based)

Zwraca

Referencja do elementu umożliwiająca modyfikację

Wyjątki

	std::logic_error	Jeśli współrzędne są poza zakresem

Definicja w linii 65 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

Odwołania w diagonalna(), diagonalna_k(), kolumna(), nad_przekatna(), odwroc(), pod_przekatna(), pokaz(), przekatna(), szachownica(), wiersz() i wstaw().

3.1.3.3 at() [2/2]

```
const int & matrix::at (
            int x,
            int y) const
```

Zwraca stałą referencję do elementu macierzy.

Parametry

	x	Indeks wiersza (0-based)
	y	Indeks kolumny (0-based)

Zwraca

Stała referencja do elementu (tylko odczyt)

Definicja w linii 77 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

3.1.3.4 diagonalna()

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

Ustawia wartości na głównej przekątnej macierzy.

Parametry

	t	Tablica zawierająca n wartości dla przekątnej

Zwraca

Referencja do bieżącej macierzy

Parametry

	t	Tablica z wartościami (wymaga n elementów)

Zwraca

Referencja do bieżącej macierzy

Definicja w linii 215 pliku matrix.cpp.

Odwołuje się do at() i n.

Odwołania w diagonalna_k() i main().

3.1.3.5 diagonalna_k()

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

Ustawia wartości na k-tej przekątnej macierzy.

Parametry

	k	Numer przekątnej (0=główna, >0=nad główną, <0=pod główną)
	t	Tablica z wartościami dla przekątnej

Zwraca

Referencja do bieżącej macierzy

Parametry

	k	Numer przekątnej (0=główna, >0=nad główną, <0=pod główną)
	t	Tablica z wartościami

Zwraca

Referencja do bieżącej macierzy

Definicja w linii 228 pliku matrix.cpp.

Odwołuje się do at(), diagonalna() i n.

Odwołania w main().

3.1.3.6 getSize()

```
int matrix::getSize () const  [inline]
```

Zwraca rozmiar macierzy.

Zwraca

Rozmiar n (dla macierzy n×n)

Definicja w linii 337 pliku matrix.h.

Odwołuje się do n.

3.1.3.7 kolumna()

```
matrix & matrix::kolumna (
            int x,
            int * t)
```

Ustawia wartości w wybranej kolumnie.

Parametry

	x	Indeks kolumny
	t	Tablica zawierająca n wartości dla kolumny

Zwraca

Referencja do bieżącej macierzy

Wyjątki

	std::logic_error	Jeśli indeks kolumny jest poza zakresem

Parametry

	x	Indeks kolumny
	t	Tablica z wartościami (wymaga n elementów)

Zwraca

Referencja do bieżącej macierzy

Wyjątki

	std::logic_error	Jeśli indeks jest poza zakresem

Definicja w linii 247 pliku matrix.cpp.

Odwołuje się do at() i n.

Odwołania w main().

3.1.3.8 losuj() [1/2]

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

Wypełnia macierz losowymi liczbami z zakresu [0, x].

Parametry

	x	Górna granica zakresu losowania

Zwraca

Referencja do bieżącej macierzy

Definicja w linii 102 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

3.1.3.9 losuj() [2/2]

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

Wypełnia macierz losowymi liczbami z zakresu [0, 9].

Zwraca

Referencja do bieżącej macierzy

Referencja do bieżącej macierzy (umożliwia łańcuchowanie wywołań)

Definicja w linii 87 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

Odwołania w main().

3.1.3.10 nad_przekatna()

```
matrix & matrix::nad_przekatna (
            void )
```

Wypełnia jedynkami obszar nad główną przekątną

Wypełnia jedynkami obszar nad główną przekątna, reszta zera.

Zwraca

Referencja do bieżącej macierzy

Definicja w linii 300 pliku matrix.cpp.

Odwołuje się do at() i n.

Odwołania w main().

3.1.3.11 odwroc()

```
matrix & matrix::odwroc (
            void )
```

Transponuje macierz (zamienia wiersze z kolumnami)

Zwraca

Referencja do bieżącej macierzy

Odbija macierz względem głównej przekątnej

Zwraca

Referencja do bieżącej macierzy

Definicja w linii 199 pliku matrix.cpp.

Odwołuje się do at() i n.

Odwołania w main().

3.1.3.12 operator()()

```
matrix & matrix::operator() (
            double d)
```

Operator wywołania - dodaje wartość do wszystkich elementów.

Operator wywołania - dodaje wartość całkowitą do wszystkich elementów.

Parametry

	d	Wartość zmiennoprzecinkowa (konwertowana na int)

Zwraca

Referencja do bieżącej macierzy

Definicja w linii 356 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

3.1.3.13 operator∗=()

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

Mnoży wszystkie elementy macierzy przez skalar.

Mnoży wszystkie elementy macierzy przez skalar (operator ∗=)

Parametry

	a	Mnożnik

Zwraca

Referencja do bieżącej macierzy

Definicja w linii 174 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

3.1.3.14 operator++()

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

Postinkrementacja - zwiększa wszystkie elementy o 1.

Zwraca

Kopia macierzy sprzed inkrementacji

Definicja w linii 185 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

3.1.3.15 operator+=()

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

Dodaje skalar do wszystkich elementów macierzy.

Dodaje skalar do wszystkich elementów macierzy (operator +=)

Parametry

	a	Wartość do dodania

Zwraca

Referencja do bieżącej macierzy

Definicja w linii 150 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

3.1.3.16 operator--()

```
matrix matrix::operator-- (
            int )
```

Postdekrementacja - zmniejsza wszystkie elementy o 1.

Zwraca

Kopia macierzy sprzed dekrementacji

Definicja w linii 342 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

3.1.3.17 operator-=()

```
matrix & matrix::operator-= (
            int a)
```

Odejmuje skalar od wszystkich elementów macierzy.

Odejmuje skalar od wszystkich elementów macierzy (operator -=)

Parametry

	a	Wartość do odjęcia

Zwraca

Referencja do bieżącej macierzy

Definicja w linii 162 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

3.1.3.18 operator<()

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

Sprawdza czy wszystkie elementy są mniejsze.

Operator mniejszości - sprawdza czy wszystkie elementy są mniejsze.

Parametry

	m	Macierz do porównania

Zwraca

true jeśli każdy element tej macierzy $<$ odpowiadający element m

Definicja w linii 395 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

3.1.3.19 operator==()

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

Sprawdza równość dwóch macierzy.

Operator równości - sprawdza czy dwie macierze są identyczne.

Parametry

	m	Macierz do porównania

Zwraca

true jeśli macierze mają ten sam rozmiar i identyczne elementy

Parametry

	m	Macierz do porównania

Zwraca

true jeśli macierze mają ten sam rozmiar i wszystkie elementy są równe

Definicja w linii 369 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

3.1.3.20 operator>()

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

Sprawdza czy wszystkie elementy są większe.

Operator większości - sprawdza czy wszystkie elementy są większe.

Parametry

	m	Macierz do porównania

Zwraca

true jeśli każdy element tej macierzy > odpowiadający element m

Definicja w linii 382 pliku matrix.cpp.

Odwołuje się do macierz_ptr i n.

3.1.3.21 pod_przekatna()

```
matrix & matrix::pod_przekatna (
            void )
```

Wypełnia jedynkami obszar pod główną przekątną

Wypełnia jedynkami obszar pod główną przekątną, reszta zera.

Zwraca

Referencja do bieżącej macierzy

Definicja w linii 287 pliku matrix.cpp.

Odwołuje się do at() i n.

Odwołania w main().

3.1.3.22 pokaz()

```
int matrix::pokaz (
            int x,
            int y) const  [inline]
```

Zwraca wartość elementu macierzy (alias dla at)

Parametry

	x	Indeks wiersza
	y	Indeks kolumny

Zwraca

Wartość elementu

Definicja w linii 92 pliku matrix.h.

Odwołuje się do at().

Odwołania w main().

3.1.3.23 przekatna()

```
matrix & matrix::przekatna (
            void )
```

Tworzy macierz jednostkową (1 na przekątnej, 0 poza)

Tworzy macierz jednostkową (jedynki na głównej przekątnej, reszta zera)

Zwraca

Referencja do bieżącej macierzy

Definicja w linii 274 pliku matrix.cpp.

Odwołuje się do at() i n.

Odwołania w main().

3.1.3.24 szachownica()

```
matrix & matrix::szachownica (
            void )
```

Wypełnia macierz wzorem szachownicy (naprzemienne 0 i 1)

Zwraca

Referencja do bieżącej macierzy

Pola gdzie (i+j) jest nieparzyste otrzymują wartość 1, pozostałe 0

Zwraca

Referencja do bieżącej macierzy

Definicja w linii 117 pliku matrix.cpp.

Odwołuje się do at() i n.

Odwołania w main().

3.1.3.25 wiersz()

```
matrix & matrix::wiersz (
            int y,
            int * t)
```

Ustawia wartości w wybranym wierszu.

Parametry

	y	Indeks wiersza
	t	Tablica zawierająca n wartości dla wiersza

Zwraca

Referencja do bieżącej macierzy

Wyjątki

	std::logic_error	Jeśli indeks wiersza jest poza zakresem

Parametry

	y	Indeks wiersza
	t	Tablica z wartościami (wymaga n elementów)

Zwraca

Referencja do bieżącej macierzy

Wyjątki

	std::logic_error	Jeśli indeks jest poza zakresem

Definicja w linii 262 pliku matrix.cpp.

Odwołuje się do at() i n.

Odwołania w main().

3.1.3.26 wstaw()

```
void matrix::wstaw (
            int x,
            int y,
            int val)  [inline]
```

Wstawia wartość do określonej pozycji w macierzy.

Parametry

	x	Indeks wiersza
	y	Indeks kolumny
	val	Wartość do wstawienia

Definicja w linii 331 pliku matrix.h.

Odwołuje się do at().

Odwołania w main().

3.1.4 Dokumentacja przyjaciół i powiązanych symboli

3.1.4.1 operator∗ [1/3]

```
matrix operator* (
            const matrix & m,
            int a)  [friend]
```

Mnoży macierz przez skalar (macierz ∗ liczba)

Parametry

	m	Macierz
	a	Mnożnik

Zwraca

Nowa macierz z pomnożonymi elementami

Definicja w linii 479 pliku matrix.cpp.

3.1.4.2 operator∗ [2/3]

```
matrix operator* (
            const matrix & m1,
            const matrix & m2)  [friend]
```

Wykonuje mnożenie macierzowe.

Parametry

	m1	Pierwsza macierz
	m2	Druga macierz

Zwraca

Nowa macierz będąca iloczynem macierzowym

Wyjątki

	std::logic_error	Jeśli macierze mają różne rozmiary

Definicja w linii 428 pliku matrix.cpp.

3.1.4.3 operator∗ [3/3]

```
matrix operator* (
            int a,
            const matrix & m)  [friend]
```

Mnoży skalar przez macierz (liczba ∗ macierz)

Parametry

	a	Mnożnik
	m	Macierz

Zwraca

Nowa macierz z pomnożonymi elementami

Definicja w linii 493 pliku matrix.cpp.

3.1.4.4 operator+ [1/3]

```
matrix operator+ (
            const matrix & m,
            int a)  [friend]
```

Dodaje skalar do macierzy (macierz + liczba)

Parametry

	m	Macierz
	a	Wartość do dodania

Zwraca

Nowa macierz z dodaną wartością do każdego elementu

Definicja w linii 451 pliku matrix.cpp.

3.1.4.5 operator+ [2/3]

```
matrix operator+ (
            const matrix & m1,
            const matrix & m2)  [friend]
```

Dodaje dwie macierze element po elemencie.

Parametry

	m1	Pierwsza macierz
	m2	Druga macierz

Zwraca

Nowa macierz będąca sumą

Wyjątki

	std::logic_error	Jeśli macierze mają różne rozmiary

Definicja w linii 410 pliku matrix.cpp.

3.1.4.6 operator+ [3/3]

```
matrix operator+ (
            int a,
            const matrix & m)  [friend]
```

Dodaje skalar do macierzy (liczba + macierz)

Parametry

	a	Wartość do dodania
	m	Macierz

Zwraca

Nowa macierz z dodaną wartością do każdego elementu

Definicja w linii 465 pliku matrix.cpp.

3.1.4.7 operator- [1/2]

```
matrix operator- (
            const matrix & m,
            int a)  [friend]
```

Odejmuje skalar od macierzy (macierz - liczba)

Parametry

	m	Macierz
	a	Wartość do odjęcia

Zwraca

Nowa macierz z odjętą wartością

Definicja w linii 503 pliku matrix.cpp.

3.1.4.8 operator- [2/2]

```
matrix operator- (
            int a,
            const matrix & m)  [friend]
```

Odejmuje macierz od skalara (liczba - macierz)

Parametry

	a	Wartość bazowa
	m	Macierz

Zwraca

Nowa macierz gdzie każdy element = a - element_macierzy

Definicja w linii 517 pliku matrix.cpp.

3.1.4.9 operator<<

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

Operator wyjścia - wypisuje macierz do strumienia.

Parametry

	o	Strumień wyjściowy
	m	Macierz do wypisania

Zwraca

Referencja do strumienia (umożliwia łańcuchowanie)

Każdy wiersz w osobnej linii, elementy oddzielone spacjami

Parametry

	o	Strumień wyjściowy
	m	Macierz do wypisania

Zwraca

Referencja do strumienia (umożliwia łańcuchowanie)

Definicja w linii 135 pliku matrix.cpp.

3.1.5 Dokumentacja atrybutów składowych

3.1.5.1 allocated_n

```
int matrix::allocated_n  [private]
```

Rozmiar zaalokowanej pamięci.

Definicja w linii 26 pliku matrix.h.

Odwołania w alokuj().

3.1.5.2 macierz_ptr

```
std::unique_ptr<int[]> matrix::macierz_ptr  [private]
```

Wskaźnik do danych macierzy (przechowywane wierszami)

Definicja w linii 27 pliku matrix.h.

Odwołania w alokuj(), at(), at(), losuj(), losuj(), matrix(), matrix(), matrix(), operator()(), operator*=(), operator++(), operator+=(), operator--(), operator-=(), operator<(), operator==() i operator>().

3.1.5.3 n

```
int matrix::n  [private]
```

Aktualny rozmiar macierzy (n×n)

Definicja w linii 25 pliku matrix.h.

Odwołania w alokuj(), at(), at(), diagonalna(), diagonalna_k(), getSize(), kolumna(), losuj(), losuj(), matrix(), matrix(), matrix(), nad_przekatna(), odwroc(), operator()(), operator*=(), operator++(), operator+=(), operator--(), operator-=(), operator<(), operator==(), operator>(), pod_przekatna(), przekatna(), szachownica() i wiersz().

Dokumentacja dla tej klasy została wygenerowana z plików:

- matrix.h
- matrix.cpp

Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku main.cpp

Program testowy dla klasy matrix.

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

Funkcje

- int main ()

  Główna funkcja programu testowego.

4.1.1 Opis szczegółowy

Program testowy dla klasy matrix.

Program przeprowadza kompletny zestaw testów funkcjonalności klasy matrix, obejmujący konstruktory, operatory, metody transformacji i wypełniania.

Definicja w pliku main.cpp.

4.1.2 Dokumentacja funkcji

4.1.2.1 main()

```
int main ()
```

Główna funkcja programu testowego.

Przeprowadza 33 testy sprawdzające wszystkie funkcjonalności klasy matrix:

- Testy konstruktorów (domyślny, parametryczny, z tablicą, kopiujący)

- Testy metod dostępu (wstaw, pokaz, at)
- Testy transformacji (odwroc, losuj, szachownica)
- Testy wzorów (przekatna, pod_przekatna, nad_przekatna)
- Testy operacji diagonalnych (diagonalna, diagonalna_k)
- Testy operacji kolumn i wierszy
- Testy operatorów arytmetycznych (+, -, ∗, +=, -=, ∗=)
- Testy operatorów inkrementacji/dekrementacji (++, –)
- Testy operatorów porównania (==, >, <)
- Testy operatora wywołania ()
- Testy alokacji pamięci

Zwraca

0 jeśli wszystkie testy zakończą się sukcesem, 1 w przypadku błędu

Definicja w linii 32 pliku main.cpp.

Odwołuje się do matrix::alokuj(), matrix::diagonalna(), matrix::diagonalna_k(), matrix::kolumna(), matrix::losuj(), matrix::nad_przekatna(), matrix::odwroc(), matrix::pod_przekatna(), matrix::pokaz(), matrix::przekatna(), matrix::szachownica(), matrix::wiersz() i matrix::wstaw().

4.2 main.cpp

Idź do dokumentacji tego pliku.

```
00001
00009 #include <iostream>
00010 #include "matrix.h"
00011
00012 using namespace std;
00013
00032 int main() {
00033     cout « "========== TESTY MACIERZY ==========" « endl « endl;
00034
00035     try {
00036         // Test 1: Konstruktory
00037         cout « "=== TEST 1: KONSTRUKTORY ===" « endl;
00038         matrix m1(3);
00039         cout « "Macierz m1 (3x3) - inicjalizacja zerami:" « endl;
00040         cout « m1 « endl;
00041
00042         // Test 2: Konstruktor z tablicą
00043         cout « "=== TEST 2: KONSTRUKTOR Z TABLICA ===" « endl;
00044         int tab[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
00045         matrix m2(3, tab);
00046         cout « "Macierz m2 (3x3) - inicjalizacja tablica:" « endl;
00047         cout « m2 « endl;
00048
00049         // Test 3: Konstruktor kopiujący
00050         cout « "=== TEST 3: KONSTRUKTOR KOPIUJACY ===" « endl;
00051         matrix m3 = m2;
00052         cout « "Macierz m3 (kopia m2):" « endl;
00053         cout « m3 « endl;
00054
00055         // Test 4: Metoda wstaw
00056         cout « "=== TEST 4: METODA WSTAW ===" « endl;
00057         m1.wstaw(0, 0, 10);
00058         m1.wstaw(1, 1, 20);
00059         m1.wstaw(2, 2, 30);
00060         cout « "Macierz po wstawieniu wartosci:" « endl;
00061         cout « m1 « endl;
00062
```

```
00063         // Test 5: Metoda pokaz
00064         cout « "=== TEST 5: METODA POKAZ ===" « endl;
00065         cout « "Element [1,1] = " « m1.pokaz(1, 1) « endl « endl;
00066
00067         // Test 6: Obrócenie macierzy
00068         cout « "=== TEST 6: OBROCENIE (TRANSPOZYCJA) ===" « endl;
00069         matrix m_obr = m2;
00070         m_obr.odwroc();
00071         cout « "Macierz m2 po obroceniu:" « endl;
00072         cout « m_obr « endl;
00073
00074         // Test 7: Losowanie
00075         cout « "=== TEST 7: LOSOWANIE ===" « endl;
00076         matrix m_los(4);
00077         m_los.losuj();
00078         cout « "Macierz 4x4 - losowe cyfry 0-9:" « endl;
00079         cout « m_los « endl;
00080
00081         // Test 8: Szachownica
00082         cout « "=== TEST 8: WZOR SZACHOWNICA ===" « endl;
00083         matrix m_szach(6);
00084         m_szach.szachownica();
00085         cout « "Szachownica 6x6:" « endl;
00086         cout « m_szach « endl;
00087
00088         // Test 9: Przekątna
00089         cout « "=== TEST 9: PRZEKATNA ===" « endl;
00090         matrix m_diag(4);
00091         m_diag.przekatna();
00092         cout « "Macierz diagonalna (1 na przekatnej, 0 poza):" « endl;
00093         cout « m_diag « endl;
00094
00095         // Test 10: Pod przekątną
00096         cout « "=== TEST 10: POD PRZEKATNA ===" « endl;
00097         matrix m_pod(4);
00098         m_pod.pod_przekatna();
00099         cout « "Macierz z jedynkami pod przekatna:" « endl;
00100         cout « m_pod « endl;
00101
00102         // Test 11: Nad przekątną
00103         cout « "=== TEST 11: NAD PRZEKATNA ===" « endl;
00104         matrix m_nad(4);
00105         m_nad.nad_przekatna();
00106         cout « "Macierz z jedynkami nad przekatna:" « endl;
00107         cout « m_nad « endl;
00108
00109         // Test 12: Diagonalna z danymi
00110         cout « "=== TEST 12: DIAGONALNA Z DANYMI ===" « endl;
00111         int dane[] = { 1, 2, 3 };
00112         matrix m_diag_dane(3);
00113         m_diag_dane.diagonalna(dane);
00114         cout « "Macierz z danymi na przekatnej:" « endl;
00115         cout « m_diag_dane « endl;
00116
00117         // Test 13: Diagonalna przesunięta
00118         cout « "=== TEST 13: DIAGONALNA PRZESUNIETA ===" « endl;
00119         matrix m_diag_k(4);
00120         int dane4[] = { 1, 2, 3 };
00121         m_diag_k.diagonalna_k(1, dane4);
00122         cout « "Macierz z danymi na przekatnej przesuniete o 1 do gory:" « endl;
00123         cout « m_diag_k « endl;
00124
00125         // Test 14: Kolumna
00126         cout « "=== TEST 14: USTAWIENIE KOLUMNY ===" « endl;
00127         matrix m_kol(3);
00128         int kolumna_dane[] = { 10, 20, 30 };
00129         m_kol.kolumna(1, kolumna_dane);
00130         cout « "Macierz z danymi w kolumnie 1:" « endl;
00131         cout « m_kol « endl;
00132
00133         // Test 15: Wiersz
00134         cout « "=== TEST 15: USTAWIENIE WIERSZA ===" « endl;
00135         matrix m_wie(3);
00136         int wiersz_dane[] = { 5, 10, 15 };
00137         m_wie.wiersz(1, wiersz_dane);
00138         cout « "Macierz z danymi w wierszu 1:" « endl;
00139         cout « m_wie « endl;
00140
00141         // Test 16: Dodawanie macierzy
00142         cout « "=== TEST 16: DODAWANIE MACIERZY (A+B) ===" « endl;
00143         matrix m4(3, tab);
00144         int tab2[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1 };
00145         matrix m5(3, tab2);
00146         matrix wynik_dodaw = m4 + m5;
00147         cout « "Macierz A:" « endl « m4 « endl;
00148         cout « "Macierz B:" « endl « m5 « endl;
00149         cout « "Wynik A + B:" « endl « wynik_dodaw « endl;
```

```
00150
00151         // Test 17: Mnożenie macierzy
00152         cout « "=== TEST 17: MNOZENIE MACIERZY (A*B) ===" « endl;
00153         int tab_a[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
00154         int tab_b[] = { 9, 8, 7, 6, 5, 4, 3, 2, 1 };
00155         matrix ma(3, tab_a);
00156         matrix mb(3, tab_b);
00157         matrix wynik_mno = ma * mb;
00158         cout « "Macierz A:" « endl « ma « endl;
00159         cout « "Macierz B:" « endl « mb « endl;
00160         cout « "Wynik A * B:" « endl « wynik_mno « endl;
00161
00162         // Test 18: Dodawanie skalara
00163         cout « "=== TEST 18: DODAWANIE SKALARA (A+5) ===" « endl;
00164         matrix m6(3, tab);
00165         matrix wynik_add_sk = m6 + 5;
00166         cout « "Macierz + 5:" « endl « wynik_add_sk « endl;
00167
00168         // Test 19: Dodawanie skalara (odwrócone)
00169         cout « "=== TEST 19: DODAWANIE SKALARA (5+A) ===" « endl;
00170         matrix wynik_add_sk2 = 5 + m6;
00171         cout « "5 + Macierz:" « endl « wynik_add_sk2 « endl;
00172
00173         // Test 20: Mnożenie skalara
00174         cout « "=== TEST 20: MNOZENIE SKALARA (A*3) ===" « endl;
00175         matrix wynik_mult_sk = m6 * 3;
00176         cout « "Macierz * 3:" « endl « wynik_mult_sk « endl;
00177
00178         // Test 21: Mnożenie skalara (odwrócone)
00179         cout « "=== TEST 21: MNOZENIE SKALARA (3*A) ===" « endl;
00180         matrix wynik_mult_sk2 = 3 * m6;
00181         cout « "3 * Macierz:" « endl « wynik_mult_sk2 « endl;
00182
00183         // Test 22: Odejmowanie skalara
00184         cout « "=== TEST 22: ODEJMOWANIE SKALARA (A-2) ===" « endl;
00185         matrix wynik_sub_sk = m6 - 2;
00186         cout « "Macierz - 2:" « endl « wynik_sub_sk « endl;
00187
00188         // Test 23: Odejmowanie skalara (odwrócone)
00189         cout « "=== TEST 23: ODEJMOWANIE SKALARA (10-A) ===" « endl;
00190         matrix wynik_sub_sk2 = 10 - m6;
00191         cout « "10 - Macierz:" « endl « wynik_sub_sk2 « endl;
00192
00193         // Test 24: Operator +=
00194         cout « "=== TEST 24: OPERATOR += ===" « endl;
00195         matrix m7(3, tab);
00196         m7 += 5;
00197         cout « "Macierz po += 5:" « endl « m7 « endl;
00198
00199         // Test 25: Operator -=
00200         cout « "=== TEST 25: OPERATOR -= ===" « endl;
00201         matrix m8(3, tab);
00202         m8 -= 2;
00203         cout « "Macierz po -= 2:" « endl « m8 « endl;
00204
00205         // Test 26: Operator *=
00206         cout « "=== TEST 26: OPERATOR *= ===" « endl;
00207         matrix m9(3, tab);
00208         m9 *= 2;
00209         cout « "Macierz po *= 2:" « endl « m9 « endl;
00210
00211         // Test 27: Inkrementacja ++
00212         cout « "=== TEST 27: INKREMENTACJA ++ ===" « endl;
00213         matrix m10(2);
00214         m10.wstaw(0, 0, 5);
00215         m10.wstaw(0, 1, 6);
00216         m10.wstaw(1, 0, 7);
00217         m10.wstaw(1, 1, 8);
00218         cout « "Przed++:" « endl « m10;
00219         m10++;
00220         cout « "Po++:" « endl « m10 « endl;
00221
00222         // Test 28: Dekrementacja --
00223         cout « "=== TEST 28: DEKREMENTACJA -- ===" « endl;
00224         matrix m11(2);
00225         m11.wstaw(0, 0, 5);
00226         m11.wstaw(0, 1, 6);
00227         m11.wstaw(1, 0, 7);
00228         m11.wstaw(1, 1, 8);
00229         cout « "Przed--:" « endl « m11;
00230         m11--;
00231         cout « "Po--:" « endl « m11 « endl;
00232
00233         // Test 29: Operator () z double
00234         cout « "=== TEST 29: OPERATOR ()(double) ===" « endl;
00235         matrix m12(2);
00236         m12.wstaw(0, 0, 5);
```

```
00237         m12.wstaw(0, 1, 6);
00238         m12.wstaw(1, 0, 7);
00239         m12.wstaw(1, 1, 8);
00240         cout « "Przed (3.7):" « endl « m12;
00241         m12(3.7);
00242         cout « "Po (3.7) - wszystkie + 3:" « endl « m12 « endl;
00243
00244         // Test 30: Operator ==
00245         cout « "=== TEST 30: OPERATOR == ===" « endl;
00246         matrix m13(2);
00247         m13.wstaw(0, 0, 1);
00248         m13.wstaw(0, 1, 2);
00249         m13.wstaw(1, 0, 3);
00250         m13.wstaw(1, 1, 4);
00251         matrix m14(2);
00252         m14.wstaw(0, 0, 1);
00253         m14.wstaw(0, 1, 2);
00254         m14.wstaw(1, 0, 3);
00255         m14.wstaw(1, 1, 4);
00256         cout « "m13 == m14? " « (m13 == m14 ? "TAK" : "NIE") « endl « endl;
00257
00258         // Test 31: Operator >
00259         cout « "=== TEST 31: OPERATOR > ===" « endl;
00260         matrix m15(2);
00261         m15.wstaw(0, 0, 5);
00262         m15.wstaw(0, 1, 5);
00263         m15.wstaw(1, 0, 5);
00264         m15.wstaw(1, 1, 5);
00265         matrix m16(2);
00266         m16.wstaw(0, 0, 2);
00267         m16.wstaw(0, 1, 2);
00268         m16.wstaw(1, 0, 2);
00269         m16.wstaw(1, 1, 2);
00270         cout « "m15 > m16? " « (m15 > m16 ? "TAK" : "NIE") « endl « endl;
00271
00272         // Test 32: Operator <
00273         cout « "=== TEST 32: OPERATOR < ===" « endl;
00274         cout « "m16 < m15? " « (m16 < m15 ? "TAK" : "NIE") « endl « endl;
00275
00276         // Test 33: Metoda alokuj
00277         cout « "=== TEST 33: METODA ALOKUJ ===" « endl;
00278         matrix m_alloc;
00279         m_alloc.alokuj(3);
00280         m_alloc.wstaw(0, 0, 10);
00281         m_alloc.wstaw(1, 1, 20);
00282         m_alloc.wstaw(2, 2, 30);
00283         cout « "Macierz po alokacji i wstawieniu danych:" « endl;
00284         cout « m_alloc « endl;
00285
00286         cout « "========== WSZYSTKIE TESTY ZAKONCZONE POMYSLNIE! ==========" « endl;
00287
00288     }
00289     catch (exception& e) {
00290         cout « "BLAD: " « e.what() « endl;
00291         return 1;
00292     }
00293
00294     return 0;
00295 }
```

4.3 Dokumentacja pliku matrix.cpp

Implementacja klasy matrix.

```
#include "matrix.h"
#include <cstdlib>
#include <ctime>
```

Funkcje

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

  Operator wyjścia - wypisuje macierz do strumienia.

- matrix operator+ (const matrix &m1, const matrix &m2)
  Dodaje dwie macierze element po elemencie.
- matrix operator∗ (const matrix &m1, const matrix &m2)
  Wykonuje mnożenie macierzowe.
- matrix operator+ (const matrix &m, int a)
  Dodaje skalar do macierzy (macierz + liczba)
- matrix operator+ (int a, const matrix &m)
  Dodaje skalar do macierzy (liczba + macierz)
- matrix operator∗ (const matrix &m, int a)
  Mnoży macierz przez skalar (macierz ∗ liczba)
- matrix operator∗ (int a, const matrix &m)
  Mnoży skalar przez macierz (liczba ∗ macierz)
- matrix operator- (const matrix &m, int a)
  Odejmuje skalar od macierzy (macierz - liczba)
- matrix operator- (int a, const matrix &m)
  Odejmuje macierz od skalara (liczba - macierz)

4.3.1 Opis szczegółowy

Implementacja klasy matrix.

Definicja w pliku matrix.cpp.

4.3.2 Dokumentacja funkcji

4.3.2.1 operator∗() [1/3]

```
matrix operator* (
            const matrix & m,
            int a)
```

Mnoży macierz przez skalar (macierz ∗ liczba)

Parametry

	m	Macierz
	a	Mnożnik

Zwraca

Nowa macierz z pomnożonymi elementami

Definicja w linii 479 pliku matrix.cpp.

4.3.2.2 operator∗() [2/3]

```
matrix operator* (
            const matrix & m1,
            const matrix & m2)
```

Wykonuje mnożenie macierzowe.

Parametry

	m1	Pierwsza macierz
	m2	Druga macierz

Zwraca

Nowa macierz będąca iloczynem macierzowym

Wyjątki

	std::logic_error	Jeśli macierze mają różne rozmiary

Definicja w linii 428 pliku matrix.cpp.

4.3.2.3 operator∗() [3/3]

```
matrix operator* (
            int a,
            const matrix & m)
```

Mnoży skalar przez macierz (liczba ∗ macierz)

Parametry

	a	Mnożnik
	m	Macierz

Zwraca

Nowa macierz z pomnożonymi elementami

Definicja w linii 493 pliku matrix.cpp.

4.3.2.4 operator+() [1/3]

```
matrix operator+ (
            const matrix & m,
            int a)
```

Dodaje skalar do macierzy (macierz + liczba)

Parametry

	m	Macierz
	a	Wartość do dodania

Zwraca

Nowa macierz z dodaną wartością do każdego elementu

Definicja w linii 451 pliku matrix.cpp.

4.3.2.5 operator+() [2/3]

```
matrix operator+ (
            const matrix & m1,
            const matrix & m2)
```

Dodaje dwie macierze element po elemencie.

Parametry

	m1	Pierwsza macierz
	m2	Druga macierz

Zwraca

Nowa macierz będąca sumą

Wyjątki

	std::logic_error	Jeśli macierze mają różne rozmiary

Definicja w linii 410 pliku matrix.cpp.

4.3.2.6 operator+() [3/3]

```
matrix operator+ (
            int a,
            const matrix & m)
```

Dodaje skalar do macierzy (liczba + macierz)

Parametry

	a	Wartość do dodania
	m	Macierz

Zwraca

Nowa macierz z dodaną wartością do każdego elementu

Definicja w linii 465 pliku matrix.cpp.

4.3.2.7 operator-() [1/2]

```
matrix operator- (
            const matrix & m,
            int a)
```

Odejmuje skalar od macierzy (macierz - liczba)

Parametry

	m	Macierz
	a	Wartość do odjęcia

Zwraca

Nowa macierz z odjętą wartością

Definicja w linii 503 pliku matrix.cpp.

4.3.2.8 operator-() [2/2]

```
matrix operator- (
            int a,
            const matrix & m)
```

Odejmuje macierz od skalara (liczba - macierz)

Parametry

	a	Wartość bazowa
	m	Macierz

Zwraca

Nowa macierz gdzie każdy element = a - element_macierzy

Definicja w linii 517 pliku matrix.cpp.

4.3.2.9 operator<<()

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

Operator wyjścia - wypisuje macierz do strumienia.

Każdy wiersz w osobnej linii, elementy oddzielone spacjami

Parametry

	o	Strumień wyjściowy
	m	Macierz do wypisania

Zwraca

Referencja do strumienia (umożliwia łańcuchowanie)

Definicja w linii 135 pliku matrix.cpp.

4.4 matrix.cpp

Idź do dokumentacji tego pliku.

```
00001
00006 #include "matrix.h"
00007 #include <cstdlib>
00008 #include <ctime>
00009
00010  // ==================== Konstruktory i destruktor ====================
00011
00015 matrix::matrix(void) : n(0), allocated_n(0), macierz_ptr(nullptr) {}
00016
00021 matrix::matrix(int n) : n(n), allocated_n(n) {
00022     macierz_ptr = std::make_unique<int[]>(n * n);
00023     for (int i = 0; i < n * n; ++i) {
00024         macierz_ptr[i] = 0;
00025     }
00026 }
00027
00033 matrix::matrix(int n, int* t) : n(n), allocated_n(n) {
00034     macierz_ptr = std::make_unique<int[]>(n * n);
00035     for (int i = 0; i < n * n; ++i) {
00036         macierz_ptr[i] = t[i];
00037     }
00038 }
00039
```

```
00044 matrix::matrix(const matrix& m) : n(m.n), allocated_n(m.allocated_n) {
00045     macierz_ptr = std::make_unique<int[]>(n * n);
00046     for (int i = 0; i < n * n; ++i) {
00047         macierz_ptr[i] = m.macierz_ptr[i];
00048     }
00049 }
00050
00054 matrix::~matrix(void) {}
00055
00056 // ==================== Metody dostępowe ====================
00057
00065 int& matrix::at(int x, int y) {
00066     if (x >= n || y >= n || x < 0 || y < 0)
00067         throw std::logic_error("Zle wspolrzedne macierzy");
00068     return macierz_ptr[x * n + y];
00069 }
00070
00077 const int& matrix::at(int x, int y) const {
00078     return macierz_ptr[x * n + y];
00079 }
00080
00081 // ==================== Metody losowania i wypełniania ====================
00082
00087 matrix& matrix::losuj(void) {
00088     static std::random_device rd;
00089     static std::mt19937 gen(rd());
00090     std::uniform_int_distribution<> dis(0, 9);
00091     for (int i = 0; i < n * n; ++i) {
00092         macierz_ptr[i] = dis(gen);
00093     }
00094     return *this;
00095 }
00096
00102 matrix& matrix::losuj(int x) {
00103     static std::random_device rd;
00104     static std::mt19937 gen(rd());
00105     std::uniform_int_distribution<> dis(0, x);
00106     for (int i = 0; i < n * n; ++i) {
00107         macierz_ptr[i] = dis(gen);
00108     }
00109     return *this;
00110 }
00111
00117 matrix& matrix::szachownica(void) {
00118     for (int i = 0; i < n; ++i) {
00119         for (int j = 0; j < n; ++j) {
00120             at(i, j) = ((i + j) % 2 != 0) ? 1 : 0;
00121         }
00122     }
00123     return *this;
00124 }
00125
00126 // ==================== Operator wyjścia ====================
00127
00135 std::ostream& operator«(std::ostream& o, const matrix& m) {
00136     for (int i = 0; i < m.n; ++i) {
00137         for (int j = 0; j < m.n; ++j) o « m.at(i, j) « " ";
00138         o « "\n";
00139     }
00140     return o;
00141 }
00142
00143 // ==================== Operatory arytmetyczne ze skalarami ====================
00144
00150 matrix& matrix::operator+=(int a) {
00151     for (int i = 0; i < n * n; ++i) {
00152         macierz_ptr[i] += a;
00153     }
00154     return *this;
00155 }
00156
00162 matrix& matrix::operator-=(int a) {
00163     for (int i = 0; i < n * n; ++i) {
00164         macierz_ptr[i] -= a;
00165     }
00166     return *this;
00167 }
00168
00174 matrix& matrix::operator*=(int a) {
00175     for (int i = 0; i < n * n; ++i) {
00176         macierz_ptr[i] *= a;
00177     }
00178     return *this;
00179 }
00180
00185 matrix matrix::operator++(int) {
00186     matrix temp(n);
```

```
00187     for (int i = 0; i < n * n; ++i) {
00188         temp.macierz_ptr[i] = macierz_ptr[i];
00189         macierz_ptr[i]++;
00190     }
00191     return temp;
00192 }
00193
00199 matrix& matrix::odwroc(void) {
00200     for (int i = 0; i < n; ++i) {
00201         for (int j = i + 1; j < n; ++j) {
00202             int temp = at(i, j);
00203             at(i, j) = at(j, i);
00204             at(j, i) = temp;
00205         }
00206     }
00207     return *this;
00208 }
00209
00215 matrix& matrix::diagonalna(int* t) {
00216     for (int i = 0; i < n; ++i) {
00217         at(i, i) = t[i];
00218     }
00219     return *this;
00220 }
00221
00228 matrix& matrix::diagonalna_k(int k, int* t) {
00229     if (k == 0) return diagonalna(t);
00230     int offset = abs(k);
00231     int limit = (k > 0) ? (n - k) : (n + k);
00232     for (int i = 0; i < limit; ++i) {
00233         int row = (k > 0) ? i : (i + offset);
00234         int col = (k > 0) ? (i + offset) : i;
00235         at(row, col) = t[i];
00236     }
00237     return *this;
00238 }
00239
00247 matrix& matrix::kolumna(int x, int* t) {
00248     if (x >= n) throw std::logic_error("Zly indeks kolumny");
00249     for (int i = 0; i < n; ++i) {
00250         at(i, x) = t[i];
00251     }
00252     return *this;
00253 }
00254
00262 matrix& matrix::wiersz(int y, int* t) {
00263     if (y >= n) throw std::logic_error("Zly indeks wiersza");
00264     for (int i = 0; i < n; ++i) {
00265         at(y, i) = t[i];
00266     }
00267     return *this;
00268 }
00269
00274 matrix& matrix::przekatna(void) {
00275     for (int i = 0; i < n; ++i) {
00276         for (int j = 0; j < n; ++j) {
00277             at(i, j) = (i == j) ? 1 : 0;
00278         }
00279     }
00280     return *this;
00281 }
00282
00287 matrix& matrix::pod_przekatna(void) {
00288     for (int i = 0; i < n; ++i) {
00289         for (int j = 0; j < n; ++j) {
00290             at(i, j) = (i > j) ? 1 : 0;
00291         }
00292     }
00293     return *this;
00294 }
00295
00300 matrix& matrix::nad_przekatna(void) {
00301     for (int i = 0; i < n; ++i) {
00302         for (int j = 0; j < n; ++j) {
00303             at(i, j) = (i < j) ? 1 : 0;
00304         }
00305     }
00306     return *this;
00307 }
00308
00320 matrix& matrix::alokuj(int rozmiar) {
00321     if (rozmiar <= 0)
00322         throw std::logic_error("Rozmiar musi byc dodatni");
00323
00324     // Brak pamięci lub za mało pamięci
00325     if (allocated_n == 0 || allocated_n < rozmiar) {
00326         macierz_ptr = std::make_unique<int[]>(rozmiar * rozmiar);
```

```
00327         allocated_n = rozmiar;
00328
00329         for (int i = 0; i < rozmiar * rozmiar; ++i)
00330             macierz_ptr[i] = 0;
00331     }
00332
00333     // ZAWSZE ustawiamy aktualny rozmiar logiczny
00334     n = rozmiar;
00335     return *this;
00336 }
00337
00342 matrix matrix::operator--(int) {
00343     matrix temp(n);
00344     for (int i = 0; i < n * n; ++i) {
00345         temp.macierz_ptr[i] = macierz_ptr[i];
00346         macierz_ptr[i]--;
00347     }
00348     return temp;
00349 }
00350
00356 matrix& matrix::operator()(double d) {
00357     int wartosc = static_cast<int>(d);
00358     for (int i = 0; i < n * n; ++i) {
00359         macierz_ptr[i] += wartosc;
00360     }
00361     return *this;
00362 }
00363
00369 bool matrix::operator==(const matrix& m) const {
00370     if (n != m.n) return false;
00371     for (int i = 0; i < n * n; ++i) {
00372         if (macierz_ptr[i] != m.macierz_ptr[i]) return false;
00373     }
00374     return true;
00375 }
00376
00382 bool matrix::operator>(const matrix& m) const {
00383     if (n != m.n) return false;
00384     for (int i = 0; i < n * n; ++i) {
00385         if (macierz_ptr[i] <= m.macierz_ptr[i]) return false;
00386     }
00387     return true;
00388 }
00389
00395 bool matrix::operator<(const matrix& m) const {
00396     if (n != m.n) return false;
00397     for (int i = 0; i < n * n; ++i) {
00398         if (macierz_ptr[i] >= m.macierz_ptr[i]) return false;
00399     }
00400     return true;
00401 }
00402
00410 matrix operator+(const matrix& m1, const matrix& m2) {
00411     if (m1.n != m2.n) {
00412         throw std::logic_error("Macierze muszą mieć ten sam rozmiar do dodawania");
00413     }
00414     matrix wynik(m1.n);
00415     for (int i = 0; i < m1.n * m1.n; ++i) {
00416         wynik.macierz_ptr[i] = m1.macierz_ptr[i] + m2.macierz_ptr[i];
00417     }
00418     return wynik;
00419 }
00420
00428 matrix operator*(const matrix& m1, const matrix& m2) {
00429     if (m1.n != m2.n) {
00430         throw std::logic_error("Macierze muszą mieć ten sam rozmiar do mnożenia");
00431     }
00432     matrix wynik(m1.n);
00433     for (int i = 0; i < m1.n; ++i) {
00434         for (int j = 0; j < m1.n; ++j) {
00435             int suma = 0;
00436             for (int k = 0; k < m1.n; ++k) {
00437                 suma += m1.at(i, k) * m2.at(k, j);
00438             }
00439             wynik.at(i, j) = suma;
00440         }
00441     }
00442     return wynik;
00443 }
00444
00451 matrix operator+(const matrix& m, int a) {
00452     matrix wynik(m.n);
00453     for (int i = 0; i < m.n * m.n; ++i) {
00454         wynik.macierz_ptr[i] = m.macierz_ptr[i] + a;
00455     }
00456     return wynik;
00457 }
```

```
00458
00465 matrix operator+(int a, const matrix& m) {
00466     matrix wynik(m.n);
00467     for (int i = 0; i < m.n * m.n; ++i) {
00468         wynik.macierz_ptr[i] = a + m.macierz_ptr[i];
00469     }
00470     return wynik;
00471 }
00472
00479 matrix operator*(const matrix& m, int a) {
00480     matrix wynik(m.n);
00481     for (int i = 0; i < m.n * m.n; ++i) {
00482         wynik.macierz_ptr[i] = m.macierz_ptr[i] * a;
00483     }
00484     return wynik;
00485 }
00486
00493 matrix operator*(int a, const matrix& m) {
00494     return m * a;
00495 }
00496
00503 matrix operator-(const matrix& m, int a) {
00504     matrix wynik(m.n);
00505     for (int i = 0; i < m.n * m.n; ++i) {
00506         wynik.macierz_ptr[i] = m.macierz_ptr[i] - a;
00507     }
00508     return wynik;
00509 }
00510
00517 matrix operator-(int a, const matrix& m) {
00518     matrix wynik(m.n);
00519     for (int i = 0; i < m.n * m.n; ++i) {
00520         wynik.macierz_ptr[i] = a - m.macierz_ptr[i];
00521     }
00522     return wynik;
00523 }
```

4.5 Dokumentacja pliku matrix.h

Deklaracja klasy matrix reprezentującej macierz kwadratową liczb całkowitych.

```
#include <iostream>
#include <memory>
#include <vector>
#include <random>
#include <iomanip>
```

Komponenty

- class matrix

  Klasa reprezentująca kwadratową macierz liczb całkowitych.

4.5.1 Opis szczegółowy

Deklaracja klasy matrix reprezentującej macierz kwadratową liczb całkowitych.

Definicja w pliku matrix.h.

4.6 matrix.h

Idź do dokumentacji tego pliku.

```
00001 #ifndef MATRIX_H
00002 #define MATRIX_H
00003
00004 #include <iostream>
00005 #include <memory>
00006 #include <vector>
00007 #include <random>
00008 #include <iomanip>
00009
00023 class matrix {
00024 private:
00025     int n;
00026     int allocated_n;
00027     std::unique_ptr<int[]> macierz_ptr;
00028
00029 public:
00030     // ==================== Konstruktory i destruktor ====================
00031
00035     matrix(void);
00036
00041     matrix(int n);
00042
00048     matrix(int n, int* t);
00049
00054     matrix(const matrix& m);
00055
00060     matrix(matrix&& other) noexcept = default;
00061
00065     ~matrix(void);
00066
00067     // ==================== Metody dostępowe ====================
00068
00076     int& at(int x, int y);
00077
00084     const int& at(int x, int y) const;
00085
00092     int pokaz(int x, int y) const { return at(x, y); }
00093
00094     // ==================== Alokacja ====================
00095
00102     matrix& alokuj(int n);
00103
00104     // ==================== Metody transformacji macierzy ====================
00105
00110     matrix& odwroc(void);
00111
00116     matrix& losuj(void);
00117
00123     matrix& losuj(int x);
00124
00130     matrix& diagonalna(int* t);
00131
00138     matrix& diagonalna_k(int k, int* t);
00139
00147     matrix& kolumna(int x, int* t);
00148
00156     matrix& wiersz(int y, int* t);
00157
00162     matrix& przekatna(void);
00163
00168     matrix& pod_przekatna(void);
00169
00174     matrix& nad_przekatna(void);
00175
00180     matrix& szachownica(void);
00181
00182     // ==================== Operatory arytmetyczne ze skalarami ====================
00183
00189     matrix& operator+=(int a);
00190
00196     matrix& operator-=(int a);
00197
00203     matrix& operator*=(int a);
00204
00209     matrix operator++(int);
00210
00215     matrix operator--(int);
00216
00222     matrix& operator()(double d);
00223
00224     // ==================== Operatory porównania ====================
00225
```

```
00231     bool operator==(const matrix& m) const;
00232
00238     bool operator>(const matrix& m) const;
00239
00245     bool operator<(const matrix& m) const;
00246
00247     // ==================== Friend operatory ====================
00248
00256     friend matrix operator+(const matrix& m1, const matrix& m2);
00257
00265     friend matrix operator*(const matrix& m1, const matrix& m2);
00266
00273     friend matrix operator+(const matrix& m, int a);
00274
00281     friend matrix operator+(int a, const matrix& m);
00282
00289     friend matrix operator*(const matrix& m, int a);
00290
00297     friend matrix operator*(int a, const matrix& m);
00298
00305     friend matrix operator-(const matrix& m, int a);
00306
00313     friend matrix operator-(int a, const matrix& m);
00314
00321     friend std::ostream& operator<<(std::ostream& o, const matrix& m);
00322
00323     // ==================== Metody pomocnicze ====================
00324
00331     void wstaw(int x, int y, int val) { at(x, y) = val; }
00332
00337     int getSize() const { return n; }
00338 };
00339 #endif
```

Skorowidz