Matrix

Wygenerowano za pomocą Doxygen 1.12.0

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 macierz kwadratową z różnymi operacjami matematycznymi.

`#include <Matrix.hpp>`

Metody publiczne

- Matrix (void)

  Konstruktor domyślny bez alokacji pamięci.
- Matrix (int n)

  Konstruktor alokujący macierz o wymiarach n x n.
- Matrix (int n, int ∗t)

  Konstruktor alokujący macierz i kopiujący dane z tabeli.
- Matrix (Matrix &m)

  Konstruktor kopiujący.
- ∼Matrix (void)

  Destruktor zwalniający pamięć.
- void alokuj (int n)

  Alokuje pamięć dla macierzy o wymiarach n x n.
- void wstaw (int x, int y, int wartosc)

  Wstawia wartość do macierzy na określonej pozycji.
- int pokaz (int x, int y)

  Zwraca wartość elementu macierzy na pozycji (x, y).
- Matrix & dowroc (void)

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

  Wypełnia macierz losowymi liczbami od 0 do 9.
- Matrix & losuj (int x)

  Wypełnia część macierzy losowymi liczbami od 0 do 9.
- Matrix & diagonalna (int ∗t)

  Tworzy macierz diagonalną z danymi z tabeli.
- Matrix & diagonalna_k (int k, int ∗t)

  Tworzy macierz diagonalną z przesuniętą przekątną.

- Matrix & kolumna (int x, int ∗t)

  Wypełnia wskazaną kolumnę danymi z tabeli.
- Matrix & wiersz (int y, int ∗t)

  Wypełnia wskazany wiersz danymi z tabeli.
- Matrix & przekatna (void)

  Tworzy macierz jednostkową.
- Matrix & pod_przekatna (void)

  Wypełnia macierz 1 poniżej przekątnej, 0 w innych miejscach.
- Matrix & nad_przekatna (void)

  Wypełnia macierz 1 powyżej przekątnej, 0 w innych miejscach.
- Matrix & szachownica (void)

  Wypełnia macierz wzorem szachownicy.
- Matrix & operator+ (Matrix &m)

  Dodaje dwie macierze.
- Matrix & operator∗ (Matrix &m)

  Mnoży dwie macierze.
- Matrix & operator+ (int a)

  Dodaje do macierzy skalar.
- Matrix & operator∗ (int a)

  Mnoży macierz przez skalar.
- Matrix & operator- (int a)

  Zmniejsza macierz o skalar.
- Matrix & operator++ (int)

  Inkrementuje każdy element macierzy o 1 (postinkrementacja).
- Matrix & operator-- (int)

  Dekrementuje każdy element macierzy o 1 (postdekrementacja).
- Matrix & operator+= (int a)

  Dodaje do każdego elementu macierzy wartość a.
- Matrix & operator-= (int a)

  Odejmuje od każdego elementu macierzy wartość a.
- Matrix & operator∗= (int a)

  Mnoży każdy element macierzy przez wartość a.
- Matrix & operator+= (double x)

  Dodaje część całkowitą liczby zmiennoprzecinkowej do każdego elementu macierzy.
- bool operator== (const Matrix &m) const

  Sprawdza, czy dwie macierze są równe.
- bool operator> (const Matrix &m)

  Sprawdza, czy wszystkie elementy macierzy są większe od odpowiadających elementów innej macierzy.
- bool operator< (const Matrix &m)

  Sprawdza, czy wszystkie elementy macierzy są mniejsze od odpowiadających elementów innej macierzy.
- bool operator!= (const Matrix &m) const

  Operator porównania nierówności macierzy.
- Matrix & operator= (const Matrix &m)

  Operator przypisania dla macierzy.
- Matrix & operator- (Matrix &m)

  Operator odejmowania dwóch macierzy.

Atrybuty prywatne

- int ∗ data
- int size

Przyjaciele

- Matrix operator+ (int scalar, const Matrix &matrix)

  Dodaje skalar do macierzy (skalar + macierz).
- Matrix & operator∗ (int scalar, const Matrix &matrix)

  Mnoży skalar przez macierz (skalar ∗ macierz).
- Matrix & operator- (int scalar, const Matrix &matrix)

  Odejmuje macierz od skalaru (skalar - macierz).
- ostream & operator<< (ostream &os, const Matrix &matrix)

  Wyświetla macierz na strumieniu wyjściowym.

3.1.1 Opis szczegółowy

Klasa reprezentująca macierz kwadratową z różnymi operacjami matematycznymi.

3.1.2 Dokumentacja konstruktora i destruktora

3.1.2.1 Matrix() [1/4]

```
Matrix::Matrix (
            void )
```

Konstruktor domyślny bez alokacji pamięci.

Domyślny konstruktor klasy Matrix.

Ustawia wskaźnik danych na nullptr i rozmiar macierzy na 0. Nie alokuje pamięci dla macierzy.

3.1.2.2 Matrix() [2/4]

```
Matrix::Matrix (
            int n)  [explicit]
```

Konstruktor alokujący macierz o wymiarach n x n.

Parametry

n	Rozmiar macierzy.

Tworzy macierz kwadratową i alokuje pamięć dla jej elementów. Wszystkie elementy są inicjalizowane wartością 0.

Parametry

n	Rozmiar macierzy (liczba wierszy i kolumn).

3.1.2.3 Matrix() [3/4]

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

Konstruktor alokujący macierz i kopiujący dane z tabeli.

Konstruktor alokujący macierz o wymiarach n x n i kopiujący dane z tabeli.

Parametry

n	Rozmiar macierzy.
t	Wskaźnik na tabelę z danymi.

Tworzy macierz kwadratową, alokuje pamięć i wypełnia ją wartościami z dostarczonej tablicy.

Parametry

n	Rozmiar macierzy (liczba wierszy i kolumn).
t	Wskaźnik na tablicę danych, która zawiera wartości do skopiowania. Tablica powinna zawierać co najmniej n ∗ n elementów.

3.1.2.4 Matrix() [4/4]

```
Matrix::Matrix (
            Matrix & m)
```

Konstruktor kopiujący.

Konstruktor kopiujący klasy Matrix.

Parametry

m	Referencja do kopiowanej macierzy.

Tworzy nową macierz, która jest kopią innej macierzy. Alokuje pamięć dla nowej macierzy i kopiuje wartości z macierzy źródłowej.

Parametry

m	Macierz, która ma zostać skopiowana.

3.1.2.5 ∼Matrix()

```
Matrix::∼Matrix (
            void )
```

Destruktor zwalniający pamięć.

Destruktor klasy Matrix.

Zwolnia pamięć zaalokowaną dla danych macierzy. Zapewnia, że zasoby są poprawnie zwolnione, aby uniknąć wycieków pamięci.

3.1.3 Dokumentacja funkcji składowych

3.1.3.1 alokuj()

```
void Matrix::alokuj (
            int n)
```

Alokuje pamięć dla macierzy o wymiarach n x n.

Alokuje pamięć dla macierzy o określonym rozmiarze.

Parametry

n	Rozmiar macierzy.

Zwraca

Referencja do bieżącego obiektu.

Parametry

n	Rozmiar macierzy (n x n).

3.1.3.2 diagonalna()

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

Tworzy macierz diagonalną z danymi z tabeli.

Ustawia macierz diagonalną na podstawie podanego wektora.

Parametry

t	Wskaźnik na tabelę z danymi.

Zwraca

Referencja do bieżącego obiektu.

Parametry

t	Tablica wartości, które mają znaleźć się na przekątnej macierzy.

Zwraca

Zwraca referencję do obiektu macierzy.

3.1.3.3 diagonalna_k()

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

Tworzy macierz diagonalną z przesuniętą przekątną.

Ustawia macierz diagonalną przesuniętą o k pozycji względem głównej przekątnej.

Parametry

k	Wartość przesunięcia przekątnej.
t	Wskaźnik na tabelę z danymi.

Zwraca

Referencja do bieżącego obiektu.

Parametry

k	Przesunięcie przekątnej względem głównej przekątnej.
t	Tablica wartości do ustawienia na przesuniętej przekątnej.

Zwraca

Zwraca referencję do obiektu macierzy.

3.1.3.4 dowroc()

```
Matrix & Matrix::dowroc (
            void )
```

Transponuje macierz (zamienia wiersze z kolumnami).

Transponuje macierz.

Zwraca

Referencja do bieżącego obiektu.

Zamienia wiersze z kolumnami macierzy.

Zwraca

Zwraca nową macierz będącą transpozycją macierzy oryginalnej.

3.1.3.5 kolumna()

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

Wypełnia wskazaną kolumnę danymi z tabeli.

Wstawia wartości do kolumny macierzy.

Parametry

x	Numer kolumny.
t	Wskaźnik na tabelę z danymi.

Zwraca

Referencja do bieżącego obiektu.

Parametry

x	Indeks kolumny do wypełnienia.
t	Tablica wartości do wstawienia w kolumnie.

Zwraca

Zwraca referencję do obiektu macierzy.

3.1.3.6 losuj() [1/2]

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

Wypełnia część macierzy losowymi liczbami od 0 do 9.

Parametry

x	Liczba losowanych elementów.

Zwraca

Referencja do bieżącego obiektu.

Liczba losowanych elementów jest określona przez parametr `x`. Losowane liczby są w przedziale [0, 9], a liczba elementów do losowania jest określona przez `x`. Elementy są losowane w losowych pozycjach macierzy.

Parametry

x	Liczba losowanych elementów.

Zwraca

Referencja do bieżącego obiektu.

3.1.3.7 losuj() [2/2]

```
Matrix & Matrix::losuj (
            void )
```

Wypełnia macierz losowymi liczbami od 0 do 9.

Zwraca

Referencja do bieżącego obiektu.

Każdy element macierzy zostaje wypełniony losową liczbą z przedziału [0, 9].

Zwraca

Referencja do bieżącego obiektu.

3.1.3.8 nad_przekatna()

```
Matrix & Matrix::nad_przekatna (
            void )
```

Wypełnia macierz 1 powyżej przekątnej, 0 w innych miejscach.

Ustawia elementy powyżej głównej przekątnej na 1, a pozostałe na 0.

Zwraca

Referencja do bieżącego obiektu.

Zwraca referencję do obiektu macierzy.

3.1.3.9 operator"!=()

```
bool Matrix::operator!= (
            const Matrix & m) const
```

Operator porównania nierówności macierzy.

Sprawdza, czy dwie macierze są różne, porównując każdy element.

Parametry

m	Macierz, z którą porównujemy bieżący obiekt.

Zwraca

true, jeśli macierze są różne, false w przeciwnym razie.

Sprawdza, czy dwie macierze są różne.

Parametry

m	Macierz, z którą porównujemy.

Zwraca

Zwraca true, jeśli macierze są różne, w przeciwnym razie false.

3.1.3.10 operator∗() [1/2]

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

Mnoży macierz przez skalar.

Operator mnożenia macierzy przez skalar.

Parametry

a	Skalar do mnożenia.

Zwraca

Wynikowa macierz.

Mnoży każdy element macierzy przez skalar.

Parametry

a	Wartość skalara.

Zwraca

Zwraca referencję do zmodyfikowanej macierzy.

3.1.3.11 operator∗() [2/2]

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

Mnoży dwie macierze.

Operator mnożenia macierzy.

Parametry

m	Macierz do mnożenia.

Zwraca

Wynikowa macierz.

Mnoży dwie macierze zgodnie z zasadami algebry macierzy.

Parametry

m	Macierz, przez którą chcemy pomnożyć.

Zwraca

Zwraca nową macierz będącą wynikiem mnożenia.

3.1.3.12 operator∗=()

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

Mnoży każdy element macierzy przez wartość a.

Parametry

a	Wartość do mnożenia.

Zwraca

Referencja do bieżącego obiektu.

3.1.3.13 operator+() [1/2]

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

Dodaje do macierzy skalar.

Operator dodawania skalaru do macierzy.

Parametry

a	Skalar do dodania.

Zwraca

Wynikowa macierz.

Dodaje skalar do każdego elementu macierzy.

Parametry

a	Wartość skalara.

Zwraca

Zwraca referencję do zmodyfikowanej macierzy.

3.1.3.14 operator+() [2/2]

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

Dodaje dwie macierze.

Operator dodawania macierzy.

Parametry

m	Macierz do dodania.

Zwraca

Wynikowa macierz.

Dodaje dwie macierze element po elemencie.

Parametry

m	Macierz, którą chcemy dodać.

Zwraca

Zwraca nową macierz będącą wynikiem dodawania.

3.1.3.15 operator++()

```
Matrix & Matrix::operator++ (
            int )
```

Inkrementuje każdy element macierzy o 1 (postinkrementacja).

Operator postinkrementacji macierzy.

Zwraca

Referencja do bieżącego obiektu.

Operator umożliwia zwiększenie każdej wartości w macierzy o 1 (postinkrementacja). Jeśli pamięć dla macierzy nie została zaalokowana, wypisywany jest komunikat o błędzie.

Ostrzeżenie

Jeśli pamięć dla macierzy nie została zaalokowana, funkcja zwróci obiekt macierzy w obecnym stanie.

Zwraca

Referencja do zaktualizowanego obiektu macierzy.

3.1.3.16 operator+=() [1/2]

```
Matrix & Matrix::operator+= (
            double x)
```

Dodaje część całkowitą liczby zmiennoprzecinkowej do każdego elementu macierzy.

Parametry

x	Liczba zmiennoprzecinkowa.

Zwraca

Referencja do bieżącego obiektu.

3.1.3.17 operator+=() [2/2]

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

Dodaje do każdego elementu macierzy wartość a.

Parametry

a	Wartość do dodania.

Zwraca

Referencja do bieżącego obiektu.

3.1.3.18 operator-() [1/2]

```
Matrix & Matrix::operator- (
            int a)
```

Zmniejsza macierz o skalar.

Operator odejmowania skalara od macierzy.

Parametry

a	Skalar do odjęcia.

Zwraca

Wynikowa macierz.

Odejmuje skalar od każdego elementu macierzy.

Parametry

a	Wartość skalara.

Zwraca

Zwraca referencję do zmodyfikowanej macierzy.

3.1.3.19 operator-() [2/2]

```
Matrix & Matrix::operator- (
            Matrix & m)
```

Operator odejmowania dwóch macierzy.

Operator odejmowania macierzy.

Tworzy nową macierz jako różnicę dwóch macierzy, odejmując elementy na odpowiadających sobie pozycjach. Obie macierze muszą mieć ten sam rozmiar.

Parametry

m	Macierz, którą odejmujemy od bieżącego obiektu.

Zwraca

Wynikowa macierz po wykonaniu operacji odejmowania.

Odejmuje dwie macierze element po elemencie.

Parametry

m	Macierz, którą chcemy odjąć.

Zwraca

Zwraca nową macierz będącą wynikiem odejmowania.

3.1.3.20 operator--()

```
Matrix & Matrix::operator-- (
            int )
```

Dekrementuje każdy element macierzy o 1 (postdekrementacja).

Operator dekrementacji postfiksowej dla macierzy.

Zwraca

Referencja do bieżącego obiektu.

Zmniejsza każdy element macierzy o 1.

Zwraca

Zwraca referencję do zmodyfikowanej macierzy.

3.1.3.21 operator-=()

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

Odejmuje od każdego elementu macierzy wartość a.

Parametry

a	Wartość do odjęcia.

Zwraca

Referencja do bieżącego obiektu.

3.1.3.22 operator<()

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

Sprawdza, czy wszystkie elementy macierzy są mniejsze od odpowiadających elementów innej macierzy.

Operator porównania "mniejszości" macierzy.

Parametry

m	Macierz do porównania.

Zwraca

true, jeśli warunek jest spełniony, false w przeciwnym razie.

Sprawdza, czy wszystkie elementy macierzy są mniejsze od odpowiadających elementów drugiej macierzy.

Parametry

m	Macierz, z którą porównujemy.

Zwraca

Zwraca true, jeśli każda wartość w tej macierzy jest mniejsza, w przeciwnym razie false.

3.1.3.23 operator=()

```
Matrix & Matrix::operator= (
            const Matrix & m)
```

Operator przypisania dla macierzy.

Operator przypisania.

Kopiuje wartości z jednej macierzy do drugiej, alokując odpowiednią pamięć. W przypadku przypisania do samego siebie operator nie wykonuje żadnych operacji.

Parametry

m	Macierz, której wartości mają zostać przypisane.

Zwraca

Referencja do bieżącego obiektu po przypisaniu.

Kopiuje wartości z jednej macierzy do drugiej.

Parametry

m	Macierz, której wartości chcemy przypisać.

Zwraca

Zwraca referencję do zmodyfikowanej macierzy.

3.1.3.24 operator==()

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

Sprawdza, czy dwie macierze są równe.

Operator porównania równości macierzy.

Parametry

m	Macierz do porównania.

Zwraca

true, jeśli macierze są równe, false w przeciwnym razie.

Sprawdza, czy dwie macierze są równe element po elemencie.

Parametry

m	Macierz, z którą porównujemy.

Zwraca

Zwraca true, jeśli macierze są równe, w przeciwnym razie false.

3.1.3.25 operator>()

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

Sprawdza, czy wszystkie elementy macierzy są większe od odpowiadających elementów innej macierzy.

Operator porównania "większości" macierzy.

Parametry

m	Macierz do porównania.

Zwraca

true, jeśli warunek jest spełniony, false w przeciwnym razie.

Sprawdza, czy wszystkie elementy macierzy są większe od odpowiadających elementów drugiej macierzy.

Parametry

m	Macierz, z którą porównujemy.

Zwraca

Zwraca true, jeśli każda wartość w tej macierzy jest większa, w przeciwnym razie false.

3.1.3.26 pod_przekatna()

```
Matrix & Matrix::pod_przekatna (
            void )
```

Wypełnia macierz 1 poniżej przekątnej, 0 w innych miejscach.

Ustawia elementy poniżej głównej przekątnej na 1, a pozostałe na 0.

Zwraca

Referencja do bieżącego obiektu.

Zwraca referencję do obiektu macierzy.

3.1.3.27 pokaz()

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

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

Pobiera wartość z macierzy w określonej pozycji.

Parametry

x	Wiersz.
y	Kolumna.

Zwraca

Wartość elementu.

Parametry

x	Indeks wiersza.
y	Indeks kolumny.

Zwraca

Zwraca wartość w macierzy na pozycji (x, y). Zwraca -1 w przypadku błędu.

3.1.3.28 przekatna()

```
Matrix & Matrix::przekatna (
            void )
```

Tworzy macierz jednostkową.

Ustawia macierz jako macierz jednostkową.

Zwraca

Referencja do bieżącego obiektu.

Zwraca referencję do obiektu macierzy.

3.1.3.29 szachownica()

```
Matrix & Matrix::szachownica (
            void )
```

Wypełnia macierz wzorem szachownicy.

Ustawia wzór szachownicy na macierzy, gdzie 0 i 1 pojawiają się naprzemiennie.

Zwraca

Referencja do bieżącego obiektu.

Zwraca referencję do obiektu macierzy.

3.1.3.30 wiersz()

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

Wypełnia wskazany wiersz danymi z tabeli.

Wstawia wartości do wiersza macierzy.

Parametry

y	Numer wiersza.
t	Wskaźnik na tabelę z danymi.

Zwraca

Referencja do bieżącego obiektu.

Parametry

y	Indeks wiersza do wypełnienia.
t	Tablica wartości do wstawienia w wierszu.

Zwraca

Zwraca referencję do obiektu macierzy.

3.1.3.31 wstaw()

```
void Matrix::wstaw (
            int x,
            int y,
            int wartosc)
```

Wstawia wartość do macierzy na określonej pozycji.

Wstawia wartość do macierzy w określonej pozycji.

Parametry

x	Wiersz.
y	Kolumna.
wartosc	Wstawiana wartość.

Zwraca

Referencja do bieżącego obiektu.

Parametry

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

3.1.4 Dokumentacja przyjaciół i powiązanych symboli

3.1.4.1 operator∗

```
Matrix & operator* (
            int scalar,
            const Matrix & matrix)  [friend]
```

Mnoży skalar przez macierz (skalar ∗ macierz).

Parametry

scalar	Skalar do mnożenia.
matrix	Macierz, którą mnożymy.

Zwraca

Wynikowa macierz.

3.1.4.2 operator+

```
Matrix operator+ (
            int scalar,
            const Matrix & matrix)  [friend]
```

Dodaje skalar do macierzy (skalar + macierz).

Parametry

scalar	Skalar do dodania.
matrix	Macierz, do której dodawany jest skalar.

Zwraca

Wynikowa macierz.

3.1.4.3 operator-

```
Matrix & operator- (
            int scalar,
            const Matrix & matrix)  [friend]
```

Odejmuje macierz od skalaru (skalar - macierz).

Parametry

scalar	Skalar, od którego odejmowana jest macierz.
matrix	Macierz do odjęcia.

Zwraca

Wynikowa macierz.

3.1.4.4 operator<<

```
ostream & operator<< (
            ostream & os,
            const Matrix & matrix)  [friend]
```

Wyświetla macierz na strumieniu wyjściowym.

Parametry

os	Strumień wyjściowy.
matrix	Macierz do wyświetlenia.

Zwraca

Strumień wyjściowy.

Wypisuje zawartość macierzy w formacie tekstowym.

Parametry

os	Strumień wyjściowy (np. std::cout).
m	Macierz do wypisania.

Zwraca

Zwraca referencję do strumienia wyjściowego.

3.1.5 Dokumentacja atrybutów składowych

3.1.5.1 data

```
int* Matrix::data  [private]
```

Wskaźnik na dane macierzy.

3.1.5.2 size

```
int Matrix::size  [private]
```

Rozmiar macierzy.

Dokumentacja dla tej klasy została wygenerowana z plików:

- Matrix.hpp
- Matrix.cpp

Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku cmake-build-debug/CMakeFiles/3.30.5/↩ CompilerIdC/CMakeCCompilerId.c

Definicje

- #define __has_include(x)
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X)
- #define STRINGIFY(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define C_STD_99 199901L
- #define C_STD_11 201112L
- #define C_STD_17 201710L
- #define C_STD_23 202311L
- #define C_VERSION

Funkcje

- int main (int argc, char ∗argv[ ])

Zmienne

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const ∗ info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const ∗ info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char ∗ info_language_standard_default
- const char ∗ info_language_extensions_default

4.1.1 Dokumentacja definicji

4.1.1.1 __has_include

```
#define __has_include(
              x)
```

Wartość:
```
0
```

4.1.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

4.1.1.3 C_STD_11

```
#define C_STD_11 201112L
```

4.1.1.4 C_STD_17

```
#define C_STD_17 201710L
```

4.1.1.5 C_STD_23

```
#define C_STD_23 202311L
```

4.1.1.6 C_STD_99

```
#define C_STD_99 199901L
```

4.1.1.7 C_VERSION

```
#define C_VERSION
```

4.1.1.8 COMPILER_ID

```
#define COMPILER_ID ""
```

4.1.1.9 DEC

```
#define DEC(
              n)
```

Wartość:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

4.1.1.10 HEX

```
#define HEX(
              n)
```

Wartość:

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

4.1.1.11 PLATFORM_ID

```
#define PLATFORM_ID
```

4.1.1.12 STRINGIFY

```
#define STRINGIFY(
              X)
```

Wartość:

```
STRINGIFY_HELPER(X)
```

4.1.1.13 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
              X)
```

Wartość:

```
#X
```

4.1.2 Dokumentacja funkcji

4.1.2.1 main()

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

4.1.3 Dokumentacja zmiennych

4.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

4.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

4.1.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Wartość początkowa:

```
= "INFO" ":" "extensions_default["



  "OFF"

"]"
```

4.1.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Wartość początkowa:

```
=
  "INFO" ":" "standard_default[" C_VERSION "]"
```

4.1.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

4.2 Dokumentacja pliku cmake-build-debug/CMakeFiles/3.30.5/↩ CompilerIdCXX/CMakeCXXCompilerId.cpp

Definicje

- #define __has_include(x)
- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X)
- #define STRINGIFY(X)
- #define PLATFORM_ID
- #define ARCHITECTURE_ID
- #define DEC(n)
- #define HEX(n)
- #define CXX_STD_98 199711L
- #define CXX_STD_11 201103L
- #define CXX_STD_14 201402L
- #define CXX_STD_17 201703L
- #define CXX_STD_20 202002L
- #define CXX_STD_23 202302L
- #define CXX_STD __cplusplus

Funkcje

- int main (int argc, char ∗argv[ ])

Zmienne

- char const ∗ info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const ∗ info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const ∗ info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char ∗ info_language_standard_default
- const char ∗ info_language_extensions_default

4.2.1 Dokumentacja definicji

4.2.1.1 __has_include

```
#define __has_include(
              x)
```

Wartość:
```
0
```

4.2.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

4.2.1.3 COMPILER_ID

```
#define COMPILER_ID ""
```

4.2.1.4 CXX_STD

```
#define CXX_STD __cplusplus
```

4.2.1.5 CXX_STD_11

```
#define CXX_STD_11 201103L
```

4.2.1.6 CXX_STD_14

```
#define CXX_STD_14 201402L
```

4.2.1.7 CXX_STD_17

```
#define CXX_STD_17 201703L
```

4.2.1.8 CXX_STD_20

```
#define CXX_STD_20 202002L
```

4.2.1.9 CXX_STD_23

```
#define CXX_STD_23 202302L
```

4.2.1.10 CXX_STD_98

```
#define CXX_STD_98 199711L
```

4.2.1.11 DEC

```
#define DEC(
              n)
```

Wartość:

```
  ('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

4.2.1.12 HEX

```
#define HEX(
              n)
```

Wartość:

```
  ('0' + ((n)»28 & 0xF)), \
  ('0' + ((n)»24 & 0xF)), \
  ('0' + ((n)»20 & 0xF)), \
  ('0' + ((n)»16 & 0xF)), \
  ('0' + ((n)»12 & 0xF)), \
  ('0' + ((n)»8  & 0xF)), \
  ('0' + ((n)»4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

4.2.1.13 PLATFORM_ID

```
#define PLATFORM_ID
```

4.2.1.14 STRINGIFY

```
#define STRINGIFY(
              X)
```

Wartość:

```
STRINGIFY_HELPER(X)
```

4.2.1.15 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(
            X)
```

Wartość:
```
#X
```

4.2.2 Dokumentacja funkcji

4.2.2.1 main()

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

4.2.3 Dokumentacja zmiennych

4.2.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

4.2.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

4.2.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Wartość początkowa:
```
= "INFO" ":" "extensions_default["



  "OFF"

"]"
```

4.2.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Wartość początkowa:
```
= "INFO" ":" "standard_default["









  "98"

"]"
```

4.2.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

4.3 Dokumentacja pliku main.cpp

Plik główny do testowania klasy Matrix.

```
#include <iostream>
#include "Matrix.hpp"
```

Funkcje

- int main ()
  Funkcja główna programu.

4.3.1 Opis szczegółowy

Plik główny do testowania klasy Matrix.

Testuje różne funkcje i operatory klasy Matrix, w tym:

- Tworzenie i inicjalizację macierzy.
- Operacje matematyczne (dodawanie, mnożenie, odejmowanie).
- Operacje na elementach macierzy (ustawianie wartości, wierszy, kolumn, przekątnych).
- Testowanie losowego wypełniania macierzy.
- Porównywanie macierzy.

4.3.2 Dokumentacja funkcji

4.3.2.1 main()

```
int main ()
```

Funkcja główna programu.

Funkcja testuje funkcjonalności klasy Matrix poprzez różnorodne operacje:

- Tworzenie macierzy, kopiowanie i modyfikowanie.
- Operacje matematyczne między macierzami oraz z wykorzystaniem skalarów.
- Losowanie wartości w macierzy i ustawianie wartości na przekątnych, wierszach i kolumnach.
- Sprawdzanie operatorów porównania, inkrementacji i dekrementacji.

Zwraca

Zwraca 0, jeśli program zakończył się poprawnie.

4.3.3 Inicjalizacja macierzy

4.3.4 Kopiowanie macierzy

4.3.5 Porównywanie macierzy

4.3.6 Operacje matematyczne

4.3.7 Operacje ze skalarami

4.3.8 Losowanie wartości w macierzy

4.3.9 Operacje na przekątnych

4.3.10 Operacje na wierszach i kolumnach

4.3.11 Wzory specjalne w macierzy

4.3.12 Testowanie inkrementacji i dekrementacji

4.4 Dokumentacja pliku Matrix.cpp

```
#include "Matrix.hpp"
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <algorithm>
```

Funkcje

- ostream & operator<< (ostream &os, const Matrix &m)

  Operator strumieniowego wypisywania macierzy.

4.4.1 Dokumentacja funkcji

4.4.1.1 operator<<()

```
ostream & operator<< (
            ostream & os,
            const Matrix & m)
```

Operator strumieniowego wypisywania macierzy.

Wyświetla macierz na strumieniu wyjściowym.

Wypisuje zawartość macierzy w formacie tekstowym.

Parametry

os	Strumień wyjściowy (np. std::cout).
m	Macierz do wypisania.

Zwraca

Zwraca referencję do strumienia wyjściowego.

4.5 Dokumentacja pliku Matrix.hpp

```
#include <iostream>
```

Komponenty

- class Matrix

  Klasa reprezentująca macierz kwadratową z różnymi operacjami matematycznymi.

4.6 Matrix.hpp

Idź do dokumentacji tego pliku.

```
00001 #include <iostream>
00002 using namespace std;
00003
00008 class Matrix {
00009 public:
00013   Matrix(void);
00014
00019   explicit Matrix(int n);
00020
00026   Matrix(int n, int* t);
00027
00032   Matrix(Matrix& m);
00033
00037   ~Matrix(void);
00038
00044   void alokuj(int n);
00045
00053    void wstaw(int x, int y, int wartosc);
00054
00061   int pokaz(int x, int y);
00062
00067   Matrix& dowroc(void);
00068
00073   Matrix& losuj(void);
00074
00080   Matrix& losuj(int x);
00081
00087   Matrix& diagonalna(int* t);
00088
00095   Matrix& diagonalna_k(int k, int* t);
00096
00103   Matrix& kolumna(int x, int* t);
00104
00111   Matrix& wiersz(int y, int* t);
00112
00117   Matrix& przekatna(void);
00118
00123   Matrix& pod_przekatna(void);
00124
00129   Matrix& nad_przekatna(void);
00130
00135   Matrix& szachownica(void);
```

```
00136
00142   Matrix& operator+(Matrix& m);
00143
00149   Matrix& operator*(Matrix& m);
00155   Matrix& operator+(int a);
00156
00162   Matrix& operator*(int a);
00163
00169   Matrix& operator-(int a);
00170
00177   friend Matrix operator+(int scalar, const Matrix& matrix);
00178
00185   friend Matrix& operator*(int scalar, const Matrix& matrix);
00186
00193   friend Matrix& operator-(int scalar, const Matrix& matrix);
00194
00199   Matrix& operator++(int);
00200
00205   Matrix& operator--(int);
00206
00212   Matrix& operator+=(int a);
00213
00219   Matrix& operator-=(int a);
00220
00226   Matrix& operator*=(int a);
00227
00233   Matrix& operator+=(double x);
00234
00241   friend ostream& operator<<(ostream& os, const Matrix& matrix);
00242
00248   bool operator==(const Matrix &m) const;
00249
00255   bool operator>(const Matrix& m);
00256
00262   bool operator<(const Matrix& m);
00263
00272   bool operator!=(const Matrix& m) const;
00273
00283   Matrix& operator=(const Matrix &m);
00284
00294   Matrix& operator-(Matrix &m);
00295
00296 private:
00297   int *data;
00298   int size;
00299 };
```

Skorowidz