matrix

Generated by Doxygen 1.12.0

Chapter 1

Programowanie zaawansowane – P4 (projekt dwuosobowy)

Napisz klasę matrix. Macierz jest kwadratowa (n na n) gdzie n jest wielkością macierzy. Macierz przechowywana jest w zmiennej dynamicznej (na stercie). Silnik biblioteki matrix należy napisać samemu bez korzystania z wyspecjalizowanych bibliotek. Funkcjonalność (metod) klasy:

- matrix(void); //konstruktor domyślny bez alokacji pamięci,
- matrix(int n); //konstruktor przeciążeniowy alokuje macierz
- wymiarach n na n,
- matrix(int n , int∗ t); //konstruktor przeciążeniowy alokuje pamięć i przepisuje dane z tabeli,
- matrix(matrix& m); //konstruktor kopiujący,
- ∼matrix(void); //destruktor,
- matrix& alokuj(int n); //jeśli macierz nie ma zaalokowanej pamięci to ją alokuje w wielkości n na n, jeśli macierz ma zaalokowaną pamięć to sprawdza czy rozmiar alokacji jest równy zdeklarowanemu rozmiarowi. W przypadku gdy tej pamięci jest mniej, pamięć ma zostać zwolniona i zaalokowana ponownie w żądanym rozmiarze. W przypadku gdy tej pamięci jest więcej pozostawić alokację bez zmian.
- matrix& wstaw(int x, int y, int wartosc); //wiersz, kolumna, wartość,
- int pokaz(int x, int y); //zwraca wartość elementu x, y,
- matrix& dowroc(void); //zamienia wiersze z kolumnami
- matrix& losuj(void); //wypełniamy cyframi od 0 do 9 wszystkie elementy macierzy
- matrix& losuj(int x); //wypełniamy cyframi od 0 do 9 elementy macierzy. Zmienna x określa ile cyfr będziemy losować. Następnie algorytm losuje, w które miejsca wstawi wylosowane cyfry,
- matrix& diagonalna(int∗ t); //po przekątnej są wpisane dane z tabeli, a pozostałe elementy są równe 0,
- matrix& diagonalna_k(int k, int∗ t); // po przekątnej są wpisane dane z tabeli, a pozostałe elementy są równe 0. Parametr k może oznaczać: 0 - przekątna przechodząca przez środek (czyli tak jak metoda diagonalna), cyfra dodatnia przesuwa diagonalną do góry macierzy o podaną cyfrę, cyfra ujemna przesuwa diagonalną w dół o podaną cyfrę,
- matrix& kolumna(int x, int∗ t); //przepisuje dane z tabeli do kolumny, którą wskazuje zmienna x,
- matrix& wiersz(int y, int∗ t); //przepisuje dane z tabeli do wiersza, który wskazuje zmienna x,
- matrix& przekatna(void); //uzupełnia macierz: 1-na przekątnej, 0-poza przekątną,
- matrix& pod_przekatna(void); //uzupełnia macierz: 1-pod przekątną, 0-nad przekątną i po przekątnej,
- matrix& nad_przekatna(void); //uzupełnia macierz: 1-nad przekątną, 0-pod przekątną i po przekątnej,

1.1 Programowanie zaawansowane – P4 (projekt dwuosobowy)

Akademia Nauk Stosowanych w Nowym Sączu 2024 / 2025r. ver. 1.0 mgr inż. Dawid Kotlarski

- matrix& szachownica(void); //uzupełnia macierz w ten sposób dla n=4:

0101

1010

0101

1010

- matrix& operator+(matrix& m); //A+B
- matrix& operator∗(matrix& m); //A∗B
- matrix& operator+(int a); //A+int
- matrix& operator∗(int a); //A∗int
- matrix& operator-(int a); //A-int
- friend matrix operator+(int a, matrix& m); //int+A
- friend matrix operator∗(int a, matrix& m); //int∗A
- friend matrix operator-(int a, matrix& m); //int-A
- matrix& operator++(int); //A++ wszystkie liczby powiększone o 1
- matrix& operator--(int); //A– wszystkie liczby pomniejszone o 1
- matrix& operator+=(int a); //każdy element w macierzy powiększamy o „a”
- matrix& operator-=(int a); //każdy element w macierzy pomniejszamy o „a”
- matrix& operator∗=(int a); //każdy element w macierzy mnożymy o „a”
- matrix& operator(double); //wszystkie cyfry są powiększone o część całkowitą z wpisanej cyfry
- friend ostream& operator<<(ostream& o, matrix& m); //wypisanie macierzy
- bool operator==(const matrix& m); //sprawdza, czy każdy element macierzy spełnia równość (,) = (,)

A=|1 2| B=|1 2|

|3 4| |3 4| jeśli nie, to nie możemy mówić, że macierze są równe,

- bool operator>(const matrix& m); //operator większości sprawdza, czy każdy element macierzy spełnia nierówność (,) > (,). Jeśli tak, to możemy powiedzieć, że macierz jest większa, w przeciwnym wypadku nie możemy stwierdzić, że macierz jest większa.
- bool operator<(const matrix& m); //tak jak wyżej tylko operator mniejszości. Na marginesie macierzy możemy nie dać rady określić, że jest równa, mniejsza i większa, wtedy mówimy że jest różna Klasa matrix musi być napisana w osobnym pliku. Funkcja main (też osobny plik) musi uruchamiać wszystkie metody celem sprawdzenia ich poprawności. Dobrym sposobem będzie wczytanie macierzy lub tabel z pliku aby nie wpisywać ich za każdym razem z klawiatury. Macierz powinna być testowana co najmniej na n=30 lub więcej. Należy zabezpieczyć program aby nie można było mnożyć różnych wielkości macierzy których matematycznie nie można pomnożyć. Celem zadania jest zapoznanie się z GitHub Copilot. Na początku należy zalogować się do GitHuba i przesłać zeskanowaną swoją legitymacje studencką (która jest podbita na ten rok

1.2 Programowanie zaawansowane – P4 (projekt dwuosobowy)

Akademia Nauk Stosowanych w Nowym Sączu 2024 / 2025r. ver. 1.0 mgr inż. Dawid Kotlarski akademicki). Po kilku dniach GitHub powinien włączyć Copilot'a za darmo. Następnie w Visual Studio lub Visual Studio Code trzeba doinstalować wtyczkę. Następnie trzeba utworzyć nowy projekt i zacząć programować. Projekt jest realizowany w dwuosobowych grupach.

W rozdziale o implementacji chcę aby pojawił się podrozdział w którym opiszecie jakie były trudności?, w czym AI sobie nie radził?, jakie popełniał błędy?, w czy sztuczna inteligencja pomogła?, może być kilka zrzutów kodu. Dodatkowo do projektu należy dołączyć dokumentację w Latex wraz z doxygenem z zastosowaniem uwag, które były omawiane na poprzednich projektach. Projekt proszę realizować bez użycia narzędzi sztucznej inteligencji takich jak chaty np.: ChatGPT. Należy tylko korzystać z GitHub Cpilot'a. Projekt jest dwuosobowy i należy pisać go równolegle. Projekt należy zapisać za pomocą oprogramowania do kontroli wersji - Git oraz wysłać projekt na GitHuba. Program napisz w języku C++.

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 matrixClass Class Reference

Klasa reprezentująca macierz o wymiarach `n x n`.

`#include <matrixClass.h>`

4.1.1 Detailed Description

Klasa reprezentująca macierz o wymiarach `n x n`.

Klasa oferuje różne funkcje do manipulacji macierzami, takie jak alokacja pamięci, wstawianie wartości, manipulacja wierszami, kolumnami, diagonalami oraz operacje arytmetyczne na macierzach.

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

- matrixClass.h

Chapter 5

File Documentation

5.1 matrix.cpp File Reference

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

Functions

- int main ()

  Główna funkcja programu.

5.1.1 Function Documentation

5.1.1.1 main()

```
int main ()
```

Główna funkcja programu.

Funkcja testuje różne funkcjonalności klasy `matrixClass`, w tym:

- konstruktory (domyślny, parametryzowany, kopiujący),
- metody manipulacji macierzą (np. `diagonalna`, `kolumna`, `wiersz`, `losuj`),
- operatory arytmetyczne i przypisania (+=, −=, ∗=) oraz operatory inkrementacji i dekrementacji.

Returns

int Kod zakończenia programu (0 w przypadku sukcesu).

Testowanie domyślnego konstruktora klasy `matrixClass`. Tworzy obiekt macierzy o domyślnych wymiarach i wartościach.

Testowanie konstruktora parametryzowanego. Tworzy obiekt macierzy o wymiarach 3x3.

Testowanie konstruktora z tablicą. Tworzy obiekt macierzy 3x3 na podstawie podanej tablicy wartości.

Testowanie konstruktora kopiującego. Tworzy obiekt `m4` jako kopię macierzy `m3`.

Testowanie metody losuj. Losowo przypisuje wartości do elementów macierzy `m2`.

Przykładowe dane do wstawienia w metodach manipulacji macierzą. Tablice dla wartości na diagonali, wierszy i kolumn.

Testowanie metody diagonalna. Ustawia wartości na głównej diagonali macierzy.

Testowanie metody diagonalna_k. Ustawia wartości na diagonali w zależności od przesunięcia.

Testowanie metody kolumna. Ustawia wartości w wybranej kolumnie macierzy.

Testowanie metody wiersz. Ustawia wartości w wybranym wierszu macierzy.

Testowanie operatora +=. Dodaje wartość 5 do każdego elementu macierzy `m2`.

Testowanie operatora $*$ (funkcja zaprzyjaźniona). Mnoży każdy element macierzy `m1` przez 3.

Testowanie operatora - (funkcja zaprzyjaźniona). Odejmuje wartość 10 od każdego elementu macierzy `m6`.

Testowanie operatora ++ (postinkrementacja). Zwiększa każdy element macierzy `m16` o 1.

Testowanie metod wstawiania wartości. Wstawia wartości do każdego elementu macierzy `m11` przy użyciu metody `wstaw`.

Testowanie operatora -=. Zmniejsza każdy element macierzy `m11` o 1.

Testowanie operatora $*=$. Mnoży każdy element macierzy `m11` przez 2.

Testowanie operatora przypisania. Ustawia wszystkie elementy macierzy `m11` na wartość 5.0.

5.2 matrix.h File Reference

5.3 matrix.h

Go to the documentation of this file.
00001

5.4 matrixClass.cpp File Reference

```
#include "matrixClass.h"
#include <iostream>
#include <cstdlib>
#include <ctime>
```

Functions

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

  Outputs the matrix to an output stream.
- matrixClass operator* (int a, const matrixClass &m)

  Multiplies each element of the matrix by a constant integer (commutative operation).
- matrixClass operator- (int a, const matrixClass &m)

  Subtracts each element of the matrix from a constant integer.
- matrixClass operator+ (int a, const matrixClass &m)

  Adds a constant integer to each element of the matrix (commutative operation).

5.4.1 Function Documentation

5.4.1.1 operator*()

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

Multiplies each element of the matrix by a constant integer (commutative operation).

Parameters

a	The integer to multiply by.
m	The matrix to multiply.

Returns

A new matrix containing the result of the multiplication.

5.4.1.2 operator+()

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

Adds a constant integer to each element of the matrix (commutative operation).

Parameters

a	The integer to add.
m	The matrix to add the constant to.

Returns

A new matrix containing the result of the addition.

5.4.1.3 operator-()

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

Subtracts each element of the matrix from a constant integer.

Parameters

a	The integer to subtract from.
m	The matrix to subtract from.

Returns

A new matrix containing the result of the subtraction.

5.4.1.4 operator<<()

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

Outputs the matrix to an output stream.

Parameters

o	The output stream.
m	The matrix to output.

Returns

The output stream with the matrix values.

5.5 matrixClass.h File Reference

```
#include <iostream>
```

5.6 matrixClass.h

Go to the documentation of this file.

```
00001 #ifndef matrixClass_H
00002 #define matrixClass_H
00003
00004 #include <iostream>
00005
00013 class matrixClass {
00014 private:
00015     int n;
00016     int** data;
00017
00018 public:
00019     // Constructors and Destructor
00020
00026     matrixClass();
00027
00035     matrixClass(int n);
00036
00045     matrixClass(int n, int* t);
00046
00054     matrixClass(const matrixClass& m);
00055
00061     ~matrixClass();
00062
00063     // Methods
00064
00073     matrixClass& alokuj(int n);
00074
00085     matrixClass& wstaw(int x, int y, int wartosc);
00086
00096     int pokaz(int x, int y);
00097
00105     matrixClass& dowroc();
00106
00114     matrixClass& losuj();
00115
00125     matrixClass& losuj(int x);
00126
00135     matrixClass& diagonalna(int* t);
00136
00146     matrixClass& diagonalna_k(int k, int* t);
00147
00157     matrixClass& kolumna(int x, int* t);
00158
00168     matrixClass& wiersz(int y, int* t);
00169
00177     matrixClass& przekatna();
00178
00186     matrixClass& pod_przekatna();
00187
00195     matrixClass& nad_przekatna();
00196
00204     matrixClass& szachownica();
00205
00206     // Operators
00207
00216     matrixClass& operator+(const matrixClass& m);
00217
00226     matrixClass& operator*(const matrixClass& m);
00227
00236     matrixClass& operator+(int a);
00237
00246     matrixClass& operator*(int a);
00247
00256     matrixClass& operator-(int a);
00257
00267     friend matrixClass operator+(int a, const matrixClass& m);
00268
00278     friend matrixClass operator*(int a, const matrixClass& m);
00279
00289     friend matrixClass operator-(int a, const matrixClass& m);
00290
00299     matrixClass& operator++(int);
00300
00309     matrixClass& operator--(int);
00310
00319     matrixClass& operator+=(int a);
00320
00329     matrixClass& operator-=(int a);
00330
00339     matrixClass& operator*=(int a);
00340
00349     matrixClass& operator=(double a);
```

```
00350
00360     friend std::ostream& operator«(std::ostream& o, const matrixClass& m);
00361
00370     bool operator==(const matrixClass& m) const;
00371
00380     bool operator>(const
```

5.7 README.md File Reference

Index