KLASA `matrix`

Klasa `matrix` powstała na potrzeby przedmiotu Metody optymalizacji i nie posiada pełnej funkcjonalności, jaką powinna posiadać klasa wykorzystywana do pracy z macierzami!

Macierz (obiekt klasy **`martix`**) reprezentowana jest przez jej wymiary **`n`** oraz **`m`** i tablicę dwuwymiarową tablicę **`M`**. Dostęp do tych zmiennych jest prywatny.

Konstruktory i destruktor:

- **`matrix(double = 0.0)`** – tworzy macierz 1x1 o podanej wartości. Jest również wykorzystywany jako **konstruktor konwertujący** (zamienia m. in. `double` i `int` na `matrix`)
- **`matrix(int, int, double = 0.0)`** – tworzy macierz nxm, której każdy element jest równy trzeciemu argumentowi
- **`matrix(int, double*)`** – tworzy macierz nx1 (wektor pionowy) i wypełnia wartościami podanymi w tablicy 1D
- **`matrix(int, int, double**)`**– tworzy macierz nxm i wypełnia wartościami podanymi w tablicy 2D
- **`matrix(const matrix&)`** – konstruktor kopiujący
- **`~matrix()`**– destruktor

Zmienna typu **`double`**, jak również zmienna typu **`int`**, jest konwertowana na macierz 1x1!

Składowe funkcje operatorowe:

- **`matrix &operator=(const matrix&)`**– operator przypisania
- **`matrix operator[](int) const`** – zwraca wskazaną kolumnę macierzy w postaci macierzy nx1 (rezultat zwracany jest przez wartość -> użycie operatora **nie jest l-wartością**)
- **`double &operator()(int = 0, int = 0)`** oraz
  `double operator()(int = 0, int = 0) const`– operatory zwracają wybrany element macierzy (mogą być wykorzystane do **konwersji `matrix` na `double`**)

Globalne funkcje operatorowe:

- **`matrix operator+(const matrix&, const matrix&)`**,
  `matrix operator-(const matrix&, const matrix&)`,
  `matrix operator*(const matrix&, const matrix&)`,
  `matrix operator/(const matrix&, const matrix&)` – operatory wykonują odpowiednie działanie pod warunkiem zgodności wymiarów macierzy
- **`matrix operator-(const matrix&)`** – macierz przeciwna
- **`bool operator<(const matrix&, const matrix&)`**,
  `bool operator>(const matrix&, const matrix&)`,
  `bool operator<=(const matrix&, const matrix&)`,
  `bool operator>=(const matrix&, const matrix&)`,
  `bool operator==(const matrix&, const matrix&)`,
  `bool operator!=(const matrix&, const matrix&)` – operatory relacji działają tylko dla macierzy 1x1

- `ostream &operator<<(ostream&, const matrix&)`– wypisanie macierzy na ekran lub do pliku csv. Podczas wypisywani elementów macierzy, część całkowita oddzielona jest od części ułamkowej za pomocą zdefiniowanego w linii 13 **SEP_SYMBOL**
- `istream &operator>>(istream&, matrix&)` – odczyt macierzy z pliku csv, txt lub klawiatury. Podając elementy macierzy należy pamiętać, że każda liczba musi kończyć się znakiem **;**Podczas wprowadzania elementów macierzy, część całkowita może być oddzielona od części ułamkowej za pomocą zdefiniowanego w linii 13 **SEP_SYMBOL** lub **.**

Funkcje składowe:

- `void set_col(const matrix&, int)` – wstawia we wskazaną kolumnę wektor pionowy (macierz nx1)
- `void set_row(const matrix&, int)` – wstawia we wskazany wiersz wektor poziomy (macierz 1xm)
- `void add_col(double = 0.0)` – dodaje do macierzy nową kolumnę i wypełnia ją podaną wartością
- `void add_row(double = 0.0)` – dodaje do macierzy nowywiersz i wypełnia go podaną wartością
- `void add_col(const matrix&)` – dodaje do macierzy nową kolumnę i wstawiaw nią podanywektor pionowy (macierz nx1)
- `void add_row(const matrix&)` – dodaje do macierzy nowywiersz i wstawie w niego podany wektor poziomu (macierz 1xm)

Funkcje globalne:

- `matrix ident_mat(int = 1)` – tworzy macierz jednostkową nxn
- `matrix rand_mat(int = 1, int = 1)` – tworzy macierz nxm i wypełnia wartościami losowymi z przedziału [0,1] o rozkładzie jednostajnym
- `matrix randn_mat(int = 1, int = 1)`– tworzy macierz nxm i wypełnia wartościami losowymi o standardowym rozkładzie normalnym
- `double m2d(const matrix&)` – zamiana macierzy 1x1 na `double`
- `double det(const matrix&)` – wyznacznik macierzy
- `matrix inv(const matrix&)` – macierz odwrotna
- `matrix trans(const matrix&)` – macierz transponowana
- `matrix pow(const matrix&, int = 2)` – podnosi macierz do potęgi
- `double norm(const matrix&)` – oblicza normę z wektora pionowego (macierzy nx1)
- `matrix hcat(const matrix&, const matrix&)` – poziome połączenie dwóch macierzy
- `matrix vcat(const matrix&, const matrix&)` – pionowe połączenie dwóch macierzy
- `matrix get_col(const matrix&, int)`–zwraca wskazaną kolumnę macierzy w postaci wektora pionowego (macierzy nx1)
- `matrix get_row(const matrix&, int)`– zwraca wskazany wiersz macierzy w postaci wektora poziomego (macierzy 1xm)

Funkcje globalne zaprzyjaźnione:

- `friend int* get_size(const matrix&)` – zwraca wymiary macierzy n oraz m
- `friend int get_len(const matrix&)` – zwraca długość wektora pionowego n (macierzy nx1)

KLASA `solution`

Rozwiązanie (obiekt klasy `solution`) jest reprezentowany przez:

- **x** – macierz (najczęściej nx1) zawierająca współrzędne punktu stanowiące rozwiązanie
- **y** – macierz (najczęściej 1x1) zawierająca wartość funkcji celu w punkcie x
- **g** – macierz nx1 zawierająca gradient funkcji celu w punkcie x
- **H** – macierz nxn zawierająca hesjan funkcji celu w punkcie x
- **ud** – macierz do dyspozycji użytkownika (used data)
- **flag** – liczba całkowita zawierająca informację o przyczynie zakończenia poszukiwania minimum funkcji celu.
- **f_calls** – zmienna statyczna zawierająca liczbę wywołań funkcji celu
- **g_calls** – zmienna statyczna zawierająca liczbę obliczeńgradientu funkcji celu
- **H_calls** – zmienna statyczna zawierająca liczbę obliczeń hesjanu funkcji celu

Dostęp do wszystkich składników klasy jest publiczny.

Konstruktory:

- **`solution(double = NAN)`**–tworzy rozwiązanie o przesłanej współrzędnej
- **`solution(const matrix&)`** – tworzy rozwiązanie o przesłanych współrzędnych
- **`solution(int, double*)`** – tworzy rozwiązanie o przesłanych współrzędnych
- **`solution(const solution&)`** – konstruktor kopiujący (macierz **ud** nie jest kopiowana jeżeli w obiekcie wzorcowym **`ud(0, 0)`** ma wartość **NAN**)

Operatory składowe i globalne:

- **`solution &operator=(const solution&)`** – operator przypisania (macierz **ud** nie jest przypisywana jeżeli w obiekcie wzorcowym **`ud(0, 0)`** ma wartość **NAN**)
- **`ostream &operator<<(ostream&, const solution&)`** – wypisanie rozwiązania na ekran (wartości **g_calls** i **H_calls** są wypisywane tylko wtedy, gdy są większe od **0**).

Funkcje składowe i globalne:

- **`static void clear_calls()`** – funkcja statyczna zerująca zmienne **f_calls**, **h_calls** oraz **H_calls**
- **`matrix fit_fun(matrix(*)(matrix, matrix, matrix), matrix = NAN, matrix = NAN)`** – funkcja wyznacza wartość zmiennej **y** poprzez wywołanie funkcji, której adres jest pierwszym argumentem oraz zwiększa **f_calls**
- **`matrix grad(matrix(*)(matrix, matrix, matrix), matrix = NAN, matrix = NAN)`** – funkcja wyznacza wartość zmiennej **g** poprzez wywołanie funkcji, której adres jest pierwszym argumentem oraz zwiększa **g_calls**
- **`matrix hess(matrix(*)(matrix, matrix, matrix), matrix = NAN, matrix = NAN)`** – funkcja wyznacza wartość zmiennej **H** poprzez wywołanie funkcji, której adres jest pierwszym argumentem oraz zwiększa **H_calls**
- **`int get_dim(const solution&)`** – funkcja zwraca długość wektora (macierzy nx1) **x**