Kolokwium/

Generated by Doxygen 1.9.1

Chapter 1

Laboratorium 2 - macierz 2∗2

1.0.1 Tresc zadan dla Panstwa (aktualniejsza jest w <tt>README.md</tt>)

Zadanie 0: absolutnie obowiazkowe, chociaz bez punktow

1. Pierwsza rzecza jest poprawa bledow kompilacji, czyli wpisanie poprawnych Panstwa danych w pliku↩ : main.cpp
2. Oddanne zadanie musi sie bezwzglednie kompilowac na systemie Linux:
   - Jesli sie nie skompiluje to jest to 0 punktow za zadanie!
   - Oczywiscie w razie problemow z kompilacja prosze się zgłaszać/pisać.
   - Dobrze, jesli nie byloby warningow kompilacji, ale za to nie obnizam punktow.
   - Aby się upewnić, że się kompiluje można skorzystać z `narzędzia Bobot`
3. Oddane zadanie nie powinno crashować na żadnym teście, jeśli crashuje proszę zrobic implementacje -fake, ktora nie dopusci do crasha nawet jesli test bedzie failowal, ale za to testy nie beda sie crashowaly. W przypadku crasha biorę pod uwagę tylko tyle testów, ile przejdzie do czasu crasha!
4. Mam program antyplagiatowy, dlatego prosze pracowac samodzielnie!
   - Osoby ktore udostepniaja swoje rozwiazania rowniez beda mialy kare!
   - Na ukaranie prowadzący ma czas 2 tygodnie po terminie oddania, czyli nawet jak ktoś otrzyma punkty wcześniej ma pewność, że za oszustwa/łatwowierność dosięgnie go niewidzialna ręka sprawiedliwości.
5. Zadanie z zalozenia bedzie sprawdzane automatycznie, przez testy jednostkowe dostepne w pliku↩ : matrixTests.cpp,
6. ∗Dobrze jakby nie bylo warningow kompilacji (flagi: `-Wall -Wextra -pedantic -Werror`, a dla hardcorów jeszcze: `-Weffc++`)
7. Punkty beda odejmowane za wycieki pamieci (jest podpiety `valgrind`)
8. Niewykluczone jest sprawdzanie reczne - zaleznie od prowadzacego dana grupe.

Tresc do implementacji - szukaj w plikach ∗.h

Chapter 2

Macierz 2x2

2.1 Przeciążanie operatorów na przykładzie Macierzy.

W zadaniu chodzi o to, aby utworzyć wygodny w użyciu typ do operacji na macierzy 2∗2, o nazwie `TwoDimensionMatrix`. Polega na tym, że mamy utworzone odpowiednie konstruktory, metody i **przeciążone operatory**.

2.1.1 Do zrobienia:

1. Zaimplementuj klasę `TwoDimensionMatrix` odzwierciedlajaca macierz 2∗2, zawierającą:
   - tablice typu `MatrixElement` (tzn. `int`, alias jest zdefiniowany w pliku `matrixElement.h`), oraz `size` (=2)
   - konstruktory:
     - bezargumentowy - zerujący wszystkie elementy
     - kopiujący
     - przyjmujący jako argument tablicę `const MatrixElement matrix[size][size]` i kopiujący z niej wartości
   - funkcja składowa do dostępu do elementów (`get()`) zwracająca odpowiedni element
   - funkcja zwracająca `size_` o nazwie (`getSize()`), proponuję aby była `static constexpr`
2. Uzupełnij klasy o następujące operacje zdefiniowane poprzez przeciążenie operatorów:
   - operator przypisania kopiujący (głęboko): `operator=()`
   - operatory wypisywania do strumienia (jako funkcja zewnętrzna) - format dowolny, byleby wszystkie elementy były w strumieniu
   - operatory wczytywania ze strumienia (jako funkcja zewnętrzna) - format dla macierzy: ``` { a, b } { c, d } ``` powinno się odbyć: ``` a b c d ```
   - operatory arytmetyczne (stosujące odpowiednie operacje na macierzach):
     - `TwoDimensionMatrix operator+(const TwoDimensionMatrix& matrix1, const TwoDimensionMatrix& matrix2); // jako funkcja globalna`
     - `TwoDimensionMatrix& operator*=(MatrixElement number); // metoda klasy`
     - Zadany operator logiczny (metoda klasy) - to jest przykład gdzie **nie** należy przeciążać operatorów: `TwoDimensionMatrix operator&&(const TwoDimensionMatrix& matrix) const;`

  - operator tablicowy dostający się po indeksie do pierwszego z wymiarów tablicy (metoda klasy), **proszę pamiętać, że mają być dwie wersje: z i bez const** `MatrixElement* operator[](size_t i);`
  - operator konwersji do `size_t`, zwracający to co `getSize()` (metoda klasy),

3. Deklaracja klasy i funkcji globalnych powinna się znaleźć w pliku "matrix.h", natomiast definicje funkcji zewnętrznych i metod klas w pliku źródłowym `matrix.cpp`

Informacje o co chodzi w paczce, na co zwrócić uwagę, jak czytać testy znajdują się w materiale `wideo`.

2.2 Uwaga:

Wszystkie atrybuty powinny być prywatne, konstruktory i metody - publiczne, metody większe niż 1-linijkowe powinny być zadeklarowane w klasie, zdefiniowane poza klasą, obiekty typów klasowych powinny być w miarę możliwości przekazywane w argumentach funkcji przez referencję, proszę też stosować słówko "const" w odpowiednich miejscach.
Mozna tworzyc dowolna ilosc metod pomocniczych, jednakze aby były one prywatne.
`Bardziej szczegółowe informacje jak pisać programy w ładnym stylu` dla zaawansowanych.

2.3 Ocenianie:

1. Ocenia `Bobot`, na ten moment w następujący sposób:
   (a) Kompilacja nadesłanego rozwiązania - bez tego zero punktów. Bobot pracuje na Linuxie, używa kompilatora g++.
   (b) Uruchamianie testów - za każdy test, który przejdzie są punkty, ale mogą być odjęte w kolejnych krokach.
   (c) Jeśli program się wywala na którymś z testów (to się pojawia często u osób pracujących na Windowsie - ten system pozwala pisać po nie-swojej pamięci, Linux nie pozwala) lub jest timeout - wtedy będzie przyznane tyle punktów ile przechodzi testów **minus dwa za karę**.
   (d) Jest odpalane narzędzie `valgrind`, które sprawdza czy umiemy obsługiwać pamięć w praktyce - jeśli nie to **minus punkt**.
   (e) Odpalane są też inne narzędzia takie jak `cppcheck`, czy `fawfinde` i inne. One nie odejmują punktów, no ale mają pomóc w pisaniu porządnych programów. Nie olewajmy tego.
   (f) Antyplagiat - za wykrycie plagiatu (jest specjalne narzędzie) otrzymuje się 0 punktów. Róbmy więc samemu!

2.4 Najczestrze pytania/błędy/problemy:

1. Jak ma działać `&&` dla macierzy?
   - Wykonująca na każdym z elementów `&&`, czyli:
     ``` { 0, 0 } { 0, 6 } { 0, 0 } {-3, 9 } && { 0, -9 } = { 0, 1 } ```
2. Jak ma działać operator tablicowy []?
   - Operator ten przyjmuje tylko jeden argument (poza this), a chcemy odnieść się w następujący sposób: `matrix[row][column]`, dlatego ten operator musi zwrócić matrix[row] typu `MatrixElement*`.
3. Mam operator indeksowania [], a kompilator jakby go nie widzi.
   - To najczęstrzy błąd w tym zadaniu - muszą być dwie wersje - jedna zwykła, a druga stała (przydomek `const`)
4. Nie rozumiem dlaczego mi test nie przechodzi!
   - Testy się **starałem** robić proste i czytelne o jak najwięcej mówiącej nazwie. Warto wejść do ciała testu i popatrzyć co się tam dzieje.
     ```

5. Użycie `chatgpt` czy można?
    - Uczyłem wiele pokoleń studentów, gdy tego nie było, dostawali się na staże, do pracy, nawet nieliczni mi dziękowali mailowo. Bez tego narzędzia są efekty. Zadania to nie tylko ich zrobienie, to też m.in. myślenie programistyczne, umiejętności, umiejętność rozwiązywania problemu, wyszukiwania informacji, przetwarzania tego co się czyta. Sztuczna inteligencja **nie pozwoli** nam na wykształcenie tak wiele, jedynie oszczędzimy czas w danej chwili (ale niekoniecznie w perspektywie całego życia).
6. Gdzie to testować?
    - W pliku `CMakeLists.txt` są tak właściwie dwa projekty - jeden zwykły, gdzie można sobie testować własną klasę ręcznie, a drugi o nazwie `tests`, który testuje testami, jakie używa Bobot.
7. Są rozbieżności między treścią `README.md`, a treściami w plikach nagłówkowych!
    - W tym roku przechodzimy na `README.md`, więc ta treść jest wiążąca.

2.5 Pytania po implementacji ćwiczenia:

1. Jaka jest różnica między przeciążaniem operatorów jako metoda klasy vs jako funkcja?
2. Których operatorów nie da się przeciążać?
3. Wymień operatory mające różną ilość argumentów?
4. Jakie konsekwencje będzie miało przeciążanie operatorów logicznych? (chodzi o lazy-evaluation)

2.6 Zadania, które warto zrobić (uwaga: nie będzie za to punktów, tylko coś cenniejszego - umiejętności)

1. Przeciążenie pozostałych operatorów.
2. Macierz 3∗3 - aby przećwiczyć zdobyte umiejętności.
3. Zrobienie zadania zgodnie z powszechnie przyjętymi standardami, `zestawienie mojego autorstwa` (można się ze mną nie zgodzić w pewnych kwestiach). Bez jakichkolwiek ostrzeżeń i warningów.
4. Poczytanie/zaimplementowanie czegoś co nas zainteresowało.
5. ∗Mając już skończonym przez nas można porównać z tym z `chatgpt`, ale dopiero po zrobieniu swojego.

2.7 Jak skonfigurować sobie pracę nad paczką:

W formie `wideo` do poprzedniej paczki (link do projektu inny, reszta analogiczna).
Alternatywnie poniżej jest to spisane w kolejnej sekcji

2.7.1 Grading (section copied from Mateusz Ślażyński, of course he agreed):

- [ ] Make sure, you have a **private** group
    - `how to create a group`
- [ ] Fork this project into your private group
    - `how to create a fork`
- [ ] Add @bobot-is-a-bot as the new project's member (role: **maintainer**)
    - `how to add an user`

2.7.2 How To Submit Solutions

1. [ ] Clone repository: `git clone` (clone only once the same repository):

   ```bash git clone <repository url> ```

2. [ ] Solve the exercises

3. [ ] Commit your changes

   ```bash git add <path to the changed files> git commit -m <commit message> ```

4. [ ] Push changes to the gitlab main branch

   ```bash git push -u origin main ```

The rest will be taken care of automatically. You can check the `GRADE.md` file for your grade / test results. Be aware that it may take some time (up to one hour) till this file. Details can be found in `./logs/` directory where You can check compilation results, tests logs etc.

2.7.3 Project Structure

```
.
 zaj2Matrix             # directory containing exercises
    CMakeLists.txt      # CMake configuration file - the file is to open out project in our IDE
    main.cpp            # main file - here we can test out solution manually, but it is not required
    matrix.h            # file to create class declaration and methods' declaration
    matrix.cpp          # file to implement methods
    matrixElement.h     # file containing type alias
    trescPdf.pdf        # documentation in PDF (generated by Doxygen)
    tests               # here are tests for exercise, inner CMakeLists.txt, GTest library used by tests
       CMakeLists.txt  # iner CMake for tests - it is included by outter CMake
       matrixTests.cpp # files with tests for exercise
       lib             # directory containing GTest library
    Doxyfile            # here is prepared file for Doxygen, to generate documentation when we type 'doxygen .
    doxyfiles           # here is logo for documentation generated by Doxygen
|    Dockerfile          # this file contains instructions how to run tests in embedded Ubuntu
|    README.md           # this file
```
```

Chapter 3

Todo List

Member FIRSTNAME

Uzupelnij swoje dane:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Class Documentation

7.1 MatrixTester Struct Reference

Inheritance diagram for MatrixTester:
Collaboration diagram for MatrixTester:

7.1.1 Detailed Description

Definition at line 25 of file matrixTests.cpp.
The documentation for this struct was generated from the following file:

- tests/matrixTests.cpp

7.2 TwoDimensionMatrix Class Reference

`#include <matrix.h>`

7.2.1 Detailed Description

Definition at line 87 of file matrix.h.
The documentation for this class was generated from the following file:

- matrix.h

Chapter 8

File Documentation

8.1 CMakeLists.txt File Reference

8.2 tests/CMakeLists.txt File Reference

Functions

- project (tests) add_subdirectory(lib) include_directories($

8.2.1 Function Documentation

8.2.1.1 project()

```
project (
            tests  )
```

Definition at line 1 of file CMakeLists.txt.

8.3 main.cpp File Reference

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

Include dependency graph for main.cpp:

8.4 matrix.cpp File Reference

```
#include <iostream>
#include <string>
#include <stdexcept>
#include <iomanip>
#include "matrix.h"
```

Include dependency graph for matrix.cpp:

8.5 matrix.h File Reference

Przeciążanie operatorów na przykładzie Macierzy (bardziej szczegolowa tresc w README.md):

```
#include <iosfwd>
#include "matrixElement.h"
```

Include dependency graph for matrix.h: This graph shows which files directly or indirectly include this file:

Classes

- class TwoDimensionMatrix

Macros

- #define UNIMPLEMENTED_CONSTRUCTORS
- #define UNIMPLEMENTED_ASIGNMENT_OPERATOR
- #define UNIMPLEMENTED_OSTREAM_OPERATOR
- #define UNIMPLEMENTED_ISTREAM_OPERATOR
- #define UNIMPLEMENTED_ARITHMETIC_OPERATORS
- #define UNIMPLEMENTED_SUBSCRIPT_OPERATOR
- #define UNIMPLEMENTED_CONVERSION_OPERATOR

8.5.1 Detailed Description

Przeciążanie operatorów na przykładzie Macierzy (bardziej szczegolowa tresc w README.md):

1. Zaimplementuj klasę TwoDimensionMatrix odzwierciedlajaca macierz 2∗2, zawierającą:
   - tablice typu MatrixElement (tzn. int), oraz size_ (=2)
   - konstruktory:
     - bezargumentowy - zerujący wszystkie elementy
     - kopiujący
     - przyjmujący jako argument tablicę (const MatrixElement matrix[size][size]) i kopiujący z niej wartości
   - funkcja składowa do dostępu do elementów (get() zwracająca odpowiedni element
   - funkcja zwracająca `size_` o nazwie `size()`, proponuję aby była `static constexpr`
   - po zaimplementowaniu usun makro `UNIMPLEMENTED_CONSTRUCTORS`
2. Uzupełnij klasy o następujące operacje zdefiniowane poprzez przeciążenie operatorów:
   - operator przypisania kopiujący (głęboko): `operator=()`
     - po zaimplementowaniu usun makro `UNIMPLEMENTED_ASIGNMENT_OPERATOR`
   - operatory wypisywania do strumienia (funkcja zewn.) - forma dowolna, byleby wszystkie elementy były w strumieniu
     - po zaimplementowaniu usun makro `UNIMPLEMENTED_OSTREAM_OPERATOR`
   - operatory wczytywania z strumienia (funkcja zewn.) - format dla macierzy: { a, b } { c, d } powinno się odbyć: ``` a b c d ``
     - `po zaimplementowaniu usun makroUNIMPLEMENTED_ISTREAM_OPERATOR`
   - `operatory arytmetyczne (stosujące odpowiednie operacje na macierzach)↩ :`
     - `TwoDimensionMatrix operator+(const TwoDimensionMatrix& matrix1, const TwoDimensionMatrix& matrix2); // jako funkcja globalna`
     - `TwoDimensionMatrix& operator*=(MatrixElement number); // metoda klasy`
     - `Zadany operator logiczny (metoda klasy): TwoDimensionMatrix operator&&(const TwoDimensionMatrix& matrix) const;`
       - `po zaimplementowaniu usun makro UNIMPLEMENTED_ARITHMETIC_↩ OPERATORS`
   - `operator tablicowy dostający się po indeksie do pierwszego z wymiarów tablicy (metoda klasy), **proszę pamiętać o wersji const** MatrixElement* operator[](size_t i);`
     - `po zaimplementowaniu usun makro UNIMPLEMENTED_SUBSCRIPT_OPERATOR`

- operator konwersji do size_t, zwracający to co size() (metoda klasy)
  - po zaimplementowaniu usun makro UNIMPLEMENTED_CONVERSION_OPERATOR Deklaracja klasy i funkcji globalnych powinna się znaleźć w pliku "matrix.h", natomiast definicje funkcji zewnętrznych i metod klas w pliku źródłowym "matrix.cpp"

8.5.1.1 Uwaga (bardziej wiazaca tresc jest w pliku <tt>README.md</tt>):

Wszystkie atrybuty powinny być prywatne, konstruktory i metody - publiczne, metody większe niż 1-linijkowe powinny być zadeklarowane w klasie, zdefiniowane poza klasą, obiekty typów klasowych powinny być w miarę możliwości przekazywane w argumentach funkcji przez referencję, proszę też stosować słówko "const" w odpowiednich miejscach.
Mozna tworzyc dowolna ilosc metod pomocniczych, jednakze aby byly one prywatne.

8.5.1.2 Punktacja:

Na maksa przejście wszystkich testów i niepoprawnych operacji na pamieci (m.in. wyciekow pamieci)

8.5.1.3 Najczestrze pytania:

1. Jak ma działać && dla macierzy? Wykonująca na każdym z elementów &&, czyli: ``` { 0, 0 } { 0, 6 } { 0, 0 } {-3, 9 } && { 0, -9 } = { 0, 1 } ```
2. Jak ma działać operator tablicowy []? Operator ten przyjmuje tylko jeden argument (poza this), a chcemy odnieść się w następujący sposób: matrix[row][column], dlatego ten operator musi zwrócić matrix[row] typu MatrixElement*.
3. Mam operator indeksowania [], a kompilator jakby go nie widzi. To najczęstrzy błąd w tym zadaniu - muszą być dwie wersje - jedna zwykła, a druga stała (przydomek const)

8.5.2 Pytania po implementacji ćwiczenia:

Note

A. Jaka jest różnica między przeciążaniem operatorów jako metoda klasy vs jako funkcja?

B. Których operatorów nie da się przeciążać?

C. Wymień operatory mające różną ilość argumentów?

D. Jakie konsekwencje będzie miało przeciążanie operatorów logicznych? (chodzi o lazy-evaluation)

8.5.3 Macro Definition Documentation

8.5.3.1 UNIMPLEMENTED_ARITHMETIC_OPERATORS

```
#define UNIMPLEMENTED_ARITHMETIC_OPERATORS
```

Definition at line 83 of file matrix.h.

8.5.3.2 UNIMPLEMENTED_ASIGNMENT_OPERATOR

```
#define UNIMPLEMENTED_ASIGNMENT_OPERATOR
```

Definition at line 80 of file matrix.h.

8.5.3.3 UNIMPLEMENTED_CONSTRUCTORS

```
#define UNIMPLEMENTED_CONSTRUCTORS
```
Definition at line 79 of file matrix.h.

8.5.3.4 UNIMPLEMENTED_CONVERSION_OPERATOR

```
#define UNIMPLEMENTED_CONVERSION_OPERATOR
```
Definition at line 85 of file matrix.h.

8.5.3.5 UNIMPLEMENTED_ISTREAM_OPERATOR

```
#define UNIMPLEMENTED_ISTREAM_OPERATOR
```
Definition at line 82 of file matrix.h.

8.5.3.6 UNIMPLEMENTED_OSTREAM_OPERATOR

```
#define UNIMPLEMENTED_OSTREAM_OPERATOR
```
Definition at line 81 of file matrix.h.

8.5.3.7 UNIMPLEMENTED_SUBSCRIPT_OPERATOR

```
#define UNIMPLEMENTED_SUBSCRIPT_OPERATOR
```
Definition at line 84 of file matrix.h.

8.6 matrixElement.h File Reference

This graph shows which files directly or indirectly include this file:

Typedefs

- using MatrixElement = int

8.6.1 Typedef Documentation

8.6.1.1 MatrixElement

```
using MatrixElement = int
```
Definition at line 4 of file matrixElement.h.

8.7 README.md File Reference

8.8 tests/matrixTests.cpp File Reference

```
#include <cstring>
#include <iostream>
#include <sstream>
#include <gtest/gtest.h>
```
Include dependency graph for matrixTests.cpp:

Classes

- struct MatrixTester

Functions

- TEST_F (MatrixTester, constructionEmptyMatrix_expectedAllElementsAreZero)
- TEST_F (MatrixTester, constructionMatrixFromTwoDimensionArray_expectedAllElementsCopied)
- TEST_F (MatrixTester, constructionMatrixFromAnotherMatrix_expectedAllElementsCopied)
- TEST_F (MatrixTester, assignmentOperatorCopyingDeeply_expectedAllElementsCopied)
- TEST_F (MatrixTester, assignmentOperatorCopyingFromItselve_expectedNotCrash)
- TEST_F (MatrixTester, checkingOstreamOperator_expectedAllElementsInStream)
- TEST_F (MatrixTester, checkingIstreamOperator_expectedAllElementsReadFromStream)
- TEST_F (MatrixTester, multiplicationMatrixMultipliedWithNumber_expectedAllElementsOfMatrixMultiplied)
- TEST_F (MatrixTester, additionOfTwoMatrixes_expectedMatrixWithSumOfElementsReturned)
- TEST_F (MatrixTester, andOfTwoMatrixes_expectedMatrixWithAllElementsAsResultOfLogicalAndReturned)
- TEST_F (MatrixTester, accessingMatrixByRowAndColumnWithIndexOperator_expectedSuccesfullAccess)
- TEST_F (MatrixTester, matrixConverseOperatorIntoSizeType_expected2ReturnElementsInEachDimentions)

8.8.1 Function Documentation

8.8.1.1 TEST_F() [1/12]

```
TEST_F (
            MatrixTester ,
            accessingMatrixByRowAndColumnWithIndexOperator_expectedSuccesfullAccess  )
```

Definition at line 278 of file matrixTests.cpp.

8.8.1.2 TEST_F() [2/12]

```
TEST_F (
            MatrixTester ,
            additionOfTwoMatrixes_expectedMatrixWithSumOfElementsReturned  )
```

Definition at line 216 of file matrixTests.cpp.

8.8.1.3 TEST_F() [3/12]

```
TEST_F (
            MatrixTester ,
            andOfTwoMatrixes_expectedMatrixWithAllElementsAsResultOfLogicalAndReturned  )
```

Definition at line 247 of file matrixTests.cpp.

8.8.1.4 TEST_F() [4/12]

```
TEST_F (
            MatrixTester ,
            assignmentOperatorCopyingDeeply_expectedAllElementsCopied  )
```

Definition at line 88 of file matrixTests.cpp.

8.8.1.5 TEST_F() [5/12]

```
TEST_F (
            MatrixTester ,
            assignmentOperatorCopyingFromItselve_expectedNotCrash  )
```

Definition at line 111 of file matrixTests.cpp.

8.8.1.6 TEST_F() [6/12]

```
TEST_F (
            MatrixTester ,
            checkingIstreamOperator_expectedAllElementsReadFromStream  )
```

Definition at line 163 of file matrixTests.cpp.

8.8.1.7 TEST_F() [7/12]

```
TEST_F (
            MatrixTester ,
            checkingOstreamOperator_expectedAllElementsInStream  )
```

Definition at line 135 of file matrixTests.cpp.

8.8.1.8 TEST_F() [8/12]

```
TEST_F (
            MatrixTester ,
            constructionEmptyMatrix_expectedAllElementsAreZero  )
```

Definition at line 29 of file matrixTests.cpp.

8.8.1.9 TEST_F() [9/12]

```
TEST_F (
            MatrixTester ,
            constructionMatrixFromAnotherMatrix_expectedAllElementsCopied  )
```

Definition at line 67 of file matrixTests.cpp.

8.8.1.10 TEST_F() [10/12]

```
TEST_F (
            MatrixTester ,
            constructionMatrixFromTwoDimensionArray_expectedAllElementsCopied  )
```

Definition at line 47 of file matrixTests.cpp.

8.8.1.11 TEST_F() [11/12]

```
TEST_F (
            MatrixTester ,
            matrixConverseOperatorIntoSizeType_expected2ReturnElementsInEachDimentions  )
```

Definition at line 304 of file matrixTests.cpp.

8.8.1.12 TEST_F() [12/12]

```
TEST_F (
            MatrixTester ,
```

```
            multiplicationMatrixMultipliedWithNumber_expectedAllElementsOfMatrixMultiplied  )
```

Definition at line 190 of file matrixTests.cpp.