My Project

Generated by Doxygen 1.9.3

Chapter 1

Laboratorium 2 - macierz 2∗2

1.0.1 Tresc zadan dla Panstwa:

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 online judge` (VPN AGH konieczny). Aby wyslac zadanie nalezy wybrac konkurs (*C++ lab1*), problem (*lab1*), oraz jezyk programowania (*c++*), prosze zalaczyc pliki matrix.h, matrix.cpp i matrixElement.h
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: matrix↩ Tests.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

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 documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 TwoDimensionMatrix Class Reference

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

- matrix.h

Chapter 5

File Documentation

5.1 matrix.h File Reference

Przeciążanie operatorów na przykładzie Macierzy:

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

Classes

- class TwoDimensionMatrix

5.1.1 Detailed Description

Przeciążanie operatorów na przykładzie Macierzy:

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 (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 (funkcja zewn.) - forma dowolna, byleby wszystkie elementy były w strumieniu

- operatory wczytywania z strumienia (funkcja zewn.) - 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): `TwoDimensionMatrix operator&&(const TwoDimensionMatrix& matrix) const;`
  - operator tablicowy dostający się po indeksie do pierwszego z wymiarów tablicy (metoda klasy), **proszę pamiętać o wersji const** `MatrixElement* operator[](size_t i);`
  - operator konwersji do size_t, zwracający to co `getSize()` (metoda klasy), Deklaracja klasy i funkcji globalnych powinna się znaleźć w pliku "matrix.h", natomiast definicje funkcji zewnętrznych i metod klas w pliku źródłowym "matrix.cpp"

5.1.1.1 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 byly one prywatne.

5.1.1.2 Punktacja:

Na maksa przejście wszystkich testów

5.1.1.3 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`)

5.1.2 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)

5.2 matrix.h

Go to the documentation of this file.

```
1 #ifndef MATRIX_H
2 #define MATRIX_H
3
67 #include <iosfwd>
68
69 #include "matrixElement.h"
70
71
72 class TwoDimensionMatrix
73 {
74     constexpr static size_t size = 2;
75
76 public:
77
78 private: // methods:
79
80 private: // fields:
81     MatrixElement matrix[size][size];
82 };
83
84 #endif // MATRIX_H
```

5.3 matrixElement.h

```
1 #ifndef MATRIX_ELEMENT
2 #define MATRIX_ELEMENT
3
4 using MatrixElement = int;
5
6 #endif // MATRIX_ELEMENT
```

Index