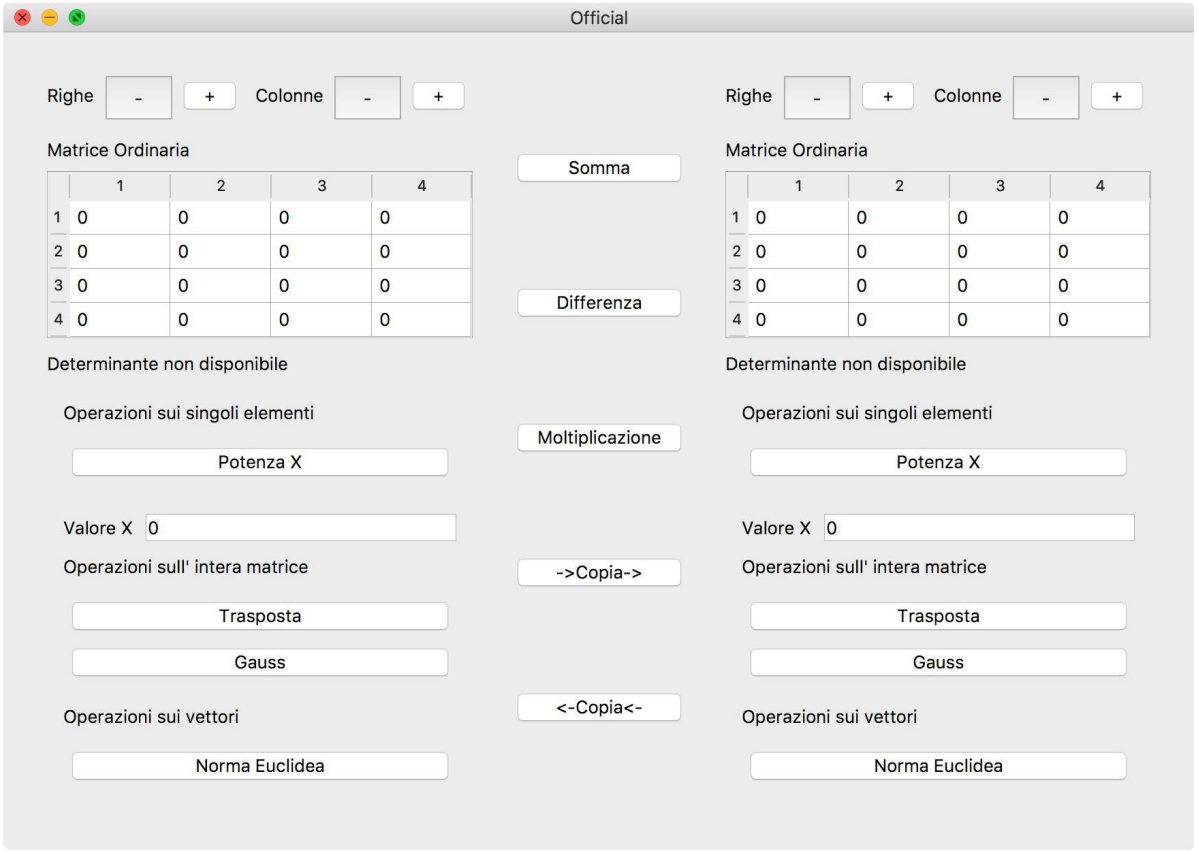
Giovanni Bergo Matricola: 1126144

OFFICIAL

Progetto di programmazione a oggetti
A.A. 2017/2018

Giovanni Bergo



Sistema Operativo di Sviluppo: macOS High Sierra 10.13.2.

Versione Qt Creator 4.5.1

Versione Qt 5.10.1

Versione QMake 4.2.1

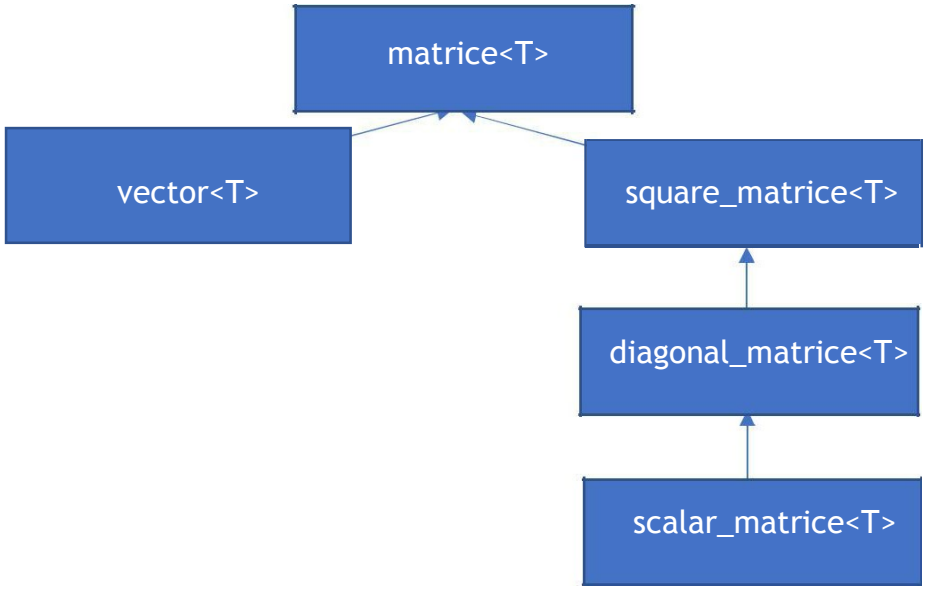
Descrizione Generale:

Official è una calcolatrice su matrici, che rende disponibili le più comuni operazioni del calcolo matriciale.

Descrizione delle Classi:

Official permette il calcolo su 5 tipi diversi di matrice, ognuno dei quali è implementato nel programma tramite una classe distinta:

- Matrice generica (matrice<T>)
- Matrice quadrata (square_matrice<T>)
- Matrice diagonale (diagonal_matrice<T>)
- Matrice scalare (scalar_matrice<T>)
- Vettore (vector<T>)



Le gerarchia delle classi è meglio descritta nella seguente rappresentazione La classe base e le sue derivate sono templatizzate, in modo da permettere l'espansione della calcolatrice a vari tipi numerici se necessario. Nel programma viene fatto uso esclusivo del tipo double. L'intera gerarchia è stata progettata per essere usata solo con **tipi numerici:** molte delle operazioni disponibili, ad esempio l'eliminazione di Gauss, non avrebbero altrimenti senso.

matrice <T>

E' la classe base della gerarchia, che incorpora tutti i principali metodi di operazioni fra matrici.

Ereditarietà
E' la classe base della gerarchia.

Attributi
l (int); Lunghezza (numero delle colonne) della matrice,
h (int); Altezza (numero di righe) della matrice,
raw_matrix (T*); Puntatore a un array, che verrà visto come una matrice di l*h elementi.

Metodi Polimorfi

~matrice<T>; Si occupa della distruzione profonda di raw_matrix.
matrix<T>* clone(); Rispetta il contratto di clonazione standard.

void Trasposta(); matrice<T> scalini (); Scalini=Gauss
I nomi sono abbastanza autoesplicativi. Marcati virtual per essere definiti in modo più efficiente nelle classi derivate.

matrice<T> operator*(const T&) const;
Metodo di moltiplicazione per uno scalare di una matrice. Marcato virtuale per essere ridefinito in modo più efficiente nelle classi derivate.

Metodi Importanti

matrice<T>& operator=(const matrice<T>&);
matrice<T>operator+(constmatrice<T>&) const;
matrice<T> operator-(const matrice<T>&) const;
bool operator==(const matrice<T>&) const;
matrice<T> operator*(const matrice<T>&) const;

Ridefinizione degli operatori più importanti.

void eleva(const double& o);

Permette di elevare al quadrato i valori della matrice secondo il valore passato per riferimento. Utilizza la libreria math.h.

square_matrice<T>

Matrice quadrata, ovvero matrice in cui "l" è uguale ad "h". Una matrice quadrata può avere un determinante, e tale proprietà è rappresentata nei metodi che la classe rende disponibili.

Ereditarietà

E' ottenuta tramite la derivazione pubblica di matrice<T>

Metodi polimorfi sovrascritti

square_matrice<T>* clone();
Sovrascritto per ritornare un puntatore di tipo corretto.

Metodi Polimorfi

T Det ()const;
Ritorna il determinante della matrice d'invocazione.

diagonal_matrice<T>

Matrice diagonale, ovvero matrice quadrata in cui tutti gli elementi che non sono sulla diagonale principale sono uguali a zero. Non rende disponibili nuove operazioni rispetto alla classe base, ma rende più efficienti quelle definite in square_matrice e matrice.

Ereditarietà

E' ottenuta tramite la derivazione pubblica di square_matrice<T>

Metodi polimorfi sovrascritti

diagonal_matrice<T>* clone();
Sovrascritto per ritornare un puntatore di tipo corretto.

diagonal_matrice<T> operator+(const diagonal_matrice<T>&) const;
diagonal_matrice<T> operator-(const diagonal_matrice<T>&) const;
diagonal_matrice<T> operator*(const diagonal_matrice<T>&) const;
virtual matrice<T> operator*(const T&);
Operatore di moltiplicazione per scalare, più efficiente in quanto agisce solo sulla diagonale.

T Det()const; void Trasposta();
Operazioni su matrici sovrascritte in modo più efficiente, tenendo in conto delle caratteristiche della matrice diagonale.

scalar_matrice<T>

Matrice scalare, ovvero matrice diagonale con tutti gli elementi sulla diagonale principale uguali fra di loro. Come per square_matrice, lo scopo è rendere i metodi già definiti più efficienti, sfruttando le proprietà di questo tipo di matrici.

Ereditarietà

E' ottenuta tramite la derivazione pubblica di diagonal_matrice<T>

Metodi polimorfi sovrascritti

scalar_matrice<T>* clone();
Sovrascritto per ritornare un puntatore di tipo corretto.

virtual T Det () const;
Sovrascritto per avere una maggior efficienza.

vector<T>

Vettore, ovvero matrice nella quale almeno una delle due dimensione è uguale a 1. Rende quindi disponibili operazioni eseguibili solo su dei vettori, come la norma euclidea.

Ereditarietà

E' ottenuta tramite la derivazione pubblica di matrice<T>

Metodi polimorfi sovrascritti

vector<T>* clone() const;
Sovrascritto per ritornare un puntatore di tipo corretto.

Metodi Importanti

T Euclidean_norm() const;
Calcola la norma euclidea del vettore di invocazione

void Normalize();
Normalizza il vettore di invocazione

Interfaccia Grafica

L'interfaccia grafica è stata implementata estendo opportunamente la classe QWidget. Lo schema in seguito chiarifica meglio l'incapsulamento dell'interfaccia.

matrice_input

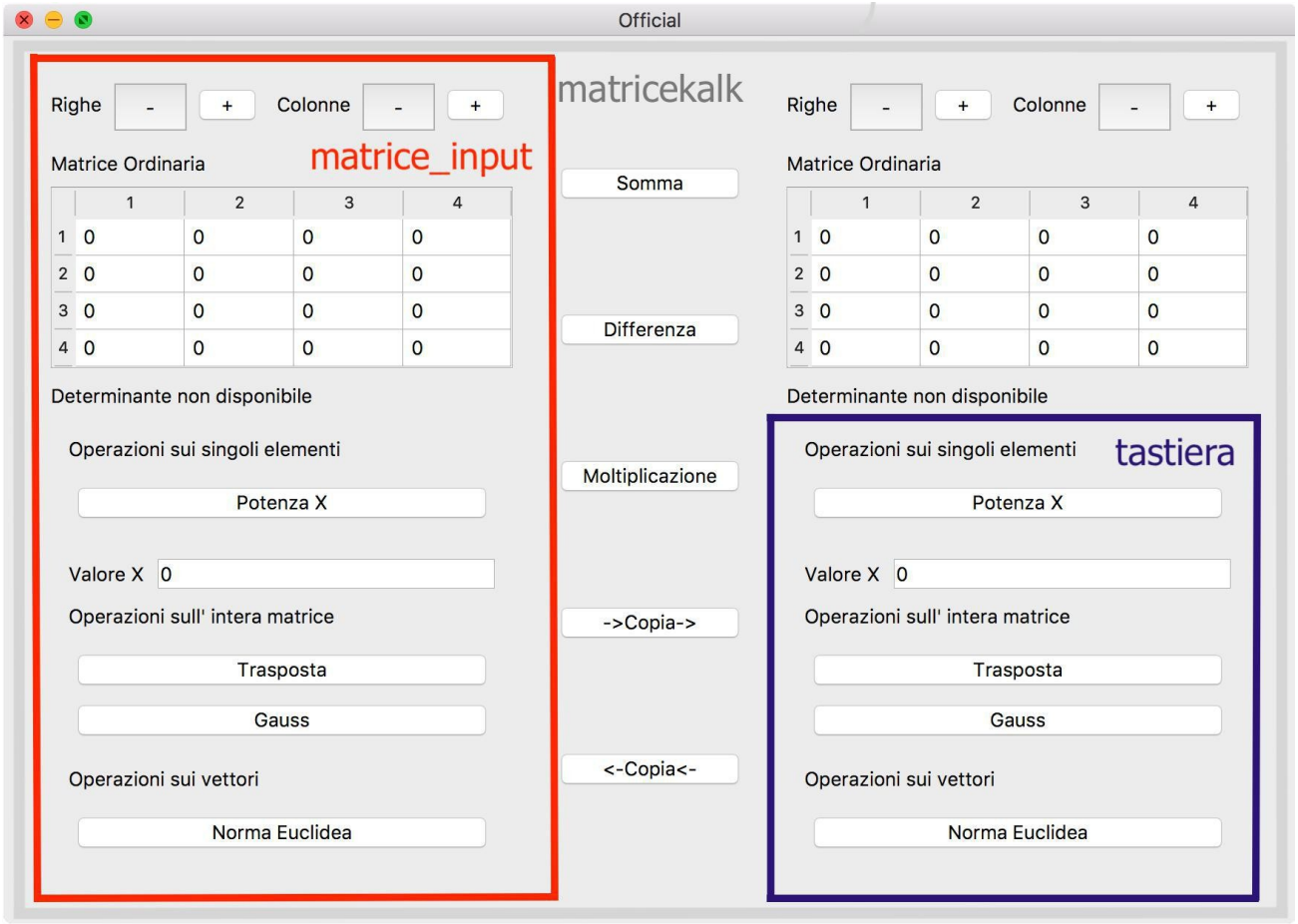
E' il principale metodo di interazione con il programma. Esso visualizza una matrice e permette di modificarla, nonché di eseguire i principali metodi messi a disposizione dalla parte logica del programma per operazioni su una sola matrice.

tastiera

Contenuta in matrice_input, permette di chiamare le operazioni su una sola matrice.

matricekalk

E' il widget principale del programma, contiene due matrice_input, più una pulsantiera centrale per eseguire operazioni fra le due matrici.



Gestione degli errori

Non sono presenti nella parte logica del programma metodi per la gestione dell'errore. L'interfaccia grafica è stata progettata in modo da non permettere operazioni non consentite. Ad esempio, non è possibile inserire valori non double in una matrice. Anche qualora l'utente provasse ad eseguire operazioni illegali, come sommare due matrici di dimensioni diverse, tali azioni vengono rifiutate direttamente dalla GUI, che segnala il ciò con un pop-up warning.

Note e scelte progettuali

Il passaggio fra i vari tipi di matrice è totalmente trasparente all'utente e automatico. E' possibile consultare il tipo attuale nell'etichetta apposita fra il selettore di dimensione e la matrice vera e propria.

Istruzioni di compilazione

La cartella principale del progetto contiene le seguenti directory:

- **Parte Logica:** Contiene la gerarchia in linguaggio C++.
- **Java:** Contiene la gerarchia in linguaggio Java.
- **Official:** Contiene tutto il codice di Qt, compreso il file .pro.

Per compilare correttamente la gui, è necessario eseguire i seguenti dalla directory principale del progetto:

```
~$cd Official
~$ qmake
~$ make
```

Per eseguire correttamente il programma, digitare da terminale:

```
~$ ./Official
```

In caso si riceva il seguente messaggio d'errore, bash: ./Official: Permesso negato è necessario spuntare "Consentire l'esecuzione del File come eseguibile" dalle proprietà del file "Official" creatosi dopo l'esecuzione di make. Per compilare correttamente la parte java, è necessario eseguire i seguenti dalla directory principale del progetto:

```
~$ cd Java
~$ javac main.java
```

Per eseguire correttamente il programma, digitare da terminale: ~$ java main.

Tempo richiesto

- Analisi preliminare e fase progettuale **15h**
- Codificazione **30h**
- Apprendimento di Qt, design e implementazione della GUI **27**