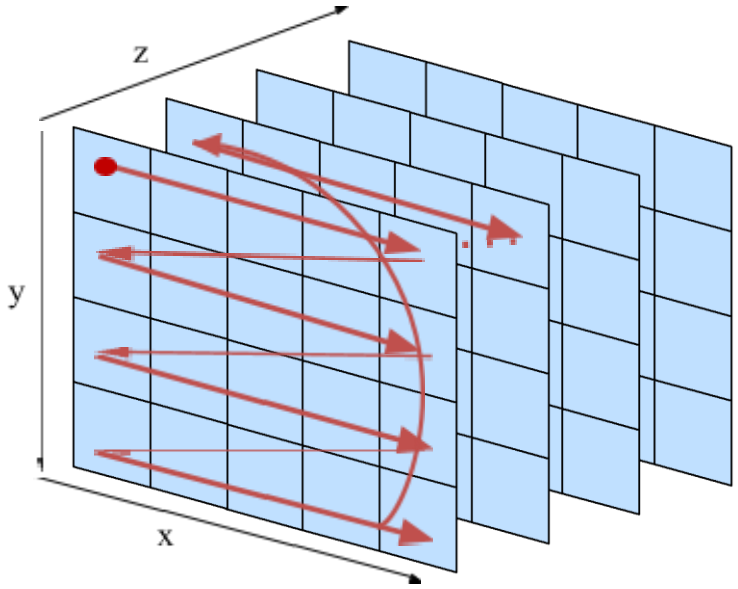
Matrice3D

Il progetto richiede la progettazione e realizzazione di una classe generica **Matrice3D** che implementa una matrice a 3 dimensioni di celle contenete dati di tipo **T**. La dimensione della matrice è scelta dall'utente in fase di costruzione dell'oggetto. Ogni cella può essere acceduta dando le coordinate del piano (z), riga (y) e colonna (x) in quest'ordine.



La classe **Matrice3D**, oltre ai metodi necessari per il suo corretto uso (come, ad esempio, conoscere il numero di righe, colonne, piani,…), dovrà avere le seguenti funzionalità:

- Deve essere possibile convertire una **Matrice3D** definita su un tipo **U** a una Matrice3D definita su un tipo **T**. Ovviamente la conversione sarà possibile se un elemento di tipo **U** è convertibile/castabile a un elemento di tipo **T**.

- Dovrà essere possibile leggere e scrivere il valore di una cella alla posizione (z,y,x) attraverso `operator()`. Es: G(1,2,3) = G(2,2,3).

- Dovranno essere implementati gli opportuni iteratori di lettura e scrittura per accedere a tutti i valori contenuti nella matrice. I valori vanno iterati secondo l'ordine indicato dalle frecce rosse nel disegno.

- Deve essere implementato un metodo, `slice` che ritorna una sotto-**Matrice3D** contenente i valori negli intervalli di coordinate z1..z2, y1..y2 e x1..x2.

- Deve essere implementato un metodo `operator==` per verificare che due oggetti **Matrice3D** contengano gli stessi valori.

- Deve essere implementato un metodo `fill` che permette di riempire una **Matrice3D** con valori presi da una sequenza di dati identificata da iteratori generici. Il riempimento deve avvenire nell'ordine di iterazione dei dati della matrice. I vecchi valori saranno sovrascritti.

Dovrà inoltre essere implementata una funzione generica `trasform` che, data una Matrice3D **A** (su tipi **T**) e un generico funtore **F**, ritorna una nuova Matrice3D **B** (su tipi **Q**) i cui elementi sono ottenuti applicando il funtore agli elementi di **A**:

$$B(i,j,k) = F(A(i,j,k))$$

Testate la classe e la funzione globale sia su tipi primitivi che su tipi custom e con diversi funtori **F**.

Utilizzare dove opportuno la gestione degli errori tramite asserzioni o eccezioni.

Nota 1: Se non indicato diversamente, nella progettazione della classe, è vietato l'uso di librerie esterne e strutture dati container della std library come std::vector, std::list e simili. E' consentito il loro uso nel codice di test nel main.

Nota 2: A parte nullptr, non potete utilizzare altri costrutti C++11 e oltre.