Corso di Laboratorio di Programmazione
A.a. 2024/25

Laboratorio 5
MyVector 2.0 con template

Esercizio

Implementare la classe `MyVector` vista nel laboratorio 4 rendendo generici gli elementi contenuti, il che significa sostituire il `double` con un tipo generico `T` utilizzando i template. Tutte le funzioni richieste nel laboratorio 4 devono essere aggiornate per tenere conto del template.

Promemoria dal laboratorio 4:
I dati devono essere gestiti tramite allocazione dinamica della memoria. Il buffer di memoria è progettato per essere più grande rispetto all'effettivo numero di elementi (come negli array parzialmente riempiti). È quindi necessario che `MyVector` tenga conto di due valori: il numero di elementi effettivamente salvati e il massimo numero di elementi gestibile con il buffer corrente. La richiesta di aggiungere elementi al buffer oltre lo spazio disponibile comporta la riallocazione di un buffer più grande, e la copia degli elementi nel nuovo buffer. È perciò importante adottare una buona politica di gestione della memoria, poiché le riallocazioni sono computazionalmente molto pesanti.

Devono essere presenti le seguenti funzioni membro:

- Le operazioni essenziali illustrate a lezione;
- Accesso ai membri tramite il doppio overloading dell'`operator[]`;
- Accesso con boundary check tramite la funzione `at()` (si veda la corrispondente funzione STL: https://www.cplusplus.com/reference/vector/vector/at/), con opportuna gestione di overloading (per lettura/scrittura) e dei casi di out of bound (eccezione);
- Funzione `push_back()` che aggiunge un elemento alla fine;
- Funzione `pop_back()` che rimuove l'ultimo elemento;
- Funzione `reserve()` (si veda la corrispondente funzione STL: https://www.cplusplus.com/reference/vector/vector/reserve/), che impone una dimensione minima del buffer (laddove la dimensinoe del buffer corrente è maggiore del valore passato da `reserve()`, non fa nulla).