Corso di Laboratorio di Programmazione

Laboratorio 5 – MyVector
22/11/2021

Implementare la classe `MyVector` che gestisce vettori di elementi generici in maniera analoga a `std::vector`. I dati devono essere gestiti tramite allocazione dinamica della memoria. Il buffer di memoria è progettato per essere più grande rispetto all'effettivo contenuto del vettore. È quindi necessario che `MyVector` tenga conto di due valori: il numero di elementi effettivamente salvati e quella del 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 lo spostamento degli elementi. È perciò importante adottare una buona politica di gestione della memoria, poiché le riallocazioni sono computazionalmente molto pesanti.

Sono presenti le seguenti funzioni membro:

- Le operazioni essenziali illustrate nella lezione 9, ove necessarie;
- Accesso ai membri tramite overloading dell'`operator[]` (vedi lezione 9);
- Accesso con boundary check tramite la funzioni `at`[1], 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`[2], che impone una dimensione minima del buffer (laddove è maggiore del valore passato da `reserve`, non fa nulla).

Nota: a vostra scelta potete implementare la classe template come descritto, oppure implementare una classe non template (con vettori di double) e passare al template in un secondo momento.

[1] Si veda la corrispondente funzione STL: https://www.cplusplus.com/reference/vector/vector/at/
[2] Si veda la corrispondente funzione STL: https://www.cplusplus.com/reference/vector/vector/reserve/