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

Laboratorio 4
MyVector

Esercizio 1

Implementare la classe `MyVector`, analoga alla classe `vector` discussa a lezione, che rappresenta vettori di `double` di lunghezza decisa in fase di costruzione (e non modificabile in seguito). Il buffer dove sono salvati i dati di `MyVector` deve essere allocato dinamicamente, poiché i dati da salvare potrebbero essere numerosi. Includere nella classe:

- un dato membro `int` che contiene la lunghezza del vettore;
- un costruttore che accetta un `int` che indica la lunghezza del vettore da costruire;
- l'overloading dell'operatore `[]` - sia in versione `const` che non `const`;
- le funzioni `safe_set()` e `safe_get()` che permettono l'accesso agli elementi del vettore effettuando il controllo di accesso entro i limiti del vettore;
- il distruttore.

Esercizio 2

Aggiornare la classe `MyVector` in modo che gestisca vettori di elementi `double` 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 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).
- I costruttori di copia e di spostamento