RELAZIONE PROGETTO C++

Aprile 2020

COGNOME: Ceciliani

NOME: Ludovico

MATRICOLA: 830288

MAIL: l.ceciliani2@campus.unimib.it

INTRODUZIONE

Il progetto consiste nell'implementazione di una coda di tipo FIFO, ovvero una queue, di valori generici che implementa i metodi fondamentali e i metodi per aggiungere un singolo elemento, rimuovere l'elemento più vecchio, verificare la presenza di un determinato elemento e permettere l'accesso di lettura e scrittura all'elemento più recenete a all'elemento più vecchio.

TIPI DI DATI

La queue è costituita da elementi implementati attraverso una struct privata definita come:

- Value = il valore dell'elemento di tipo T generico;
- Next = puntatore all'elemento successivo.

La queue è considerata come un array dove l'elemento in prima posizione è il più vecchio, mentre l'ultimo elemento è l'ultimo inserito.

IMPLEMENTAZIONE DELLA QUEUE

La queue è una classe templata che viene identificata da due puntatori, una al primo elemento e l'altro all'ultimo elemento, e la grandezza della coda stessa. Inoltre richiede un funtore di uguaglianza (==).

Come ogni classe, abbiamo bisogno dei metodi fondamentali e costruttori secondari:

- Costruttore di default, che setta i due puntatori a nullptr e la grandezza a 0;
- Costruttore di copia, che, data una queue come parametro, copia, se possibile, gli elementi di essa. Utilizza il metodo enqueue;
- Costruttore secondario templato, che permette di riempire la coda dando una sequenza di dati identificata da due iteratori;
- Distruttore, che elimina tutti gli elementi creati e allocati in memoria aiutandosi col metodo clear_helper.

METODI DELLA QUEUE

- clear = si utilizza principalmente nel distruttore, ma può anche essere utilizzato nel caso in cui si voglia svuotare il contenuto della queue.
- size = ritorna l'effettiva grandezza della queue.
- find = controlla se è presente nella queue un determinato valore. Sfrutta il funtore per l'uguaglianza.
- enqueue = inserisce nella queue un elemento con un determinato valore T. L'elemento viene inserito all'ultima posizione. Sono possibili dati duplicati.
- dequeue = rimozione del primo elemento inserito nella queue. Nel caso lo si utilizzi su una queue vuota restituisce un messaggio di errore.
- getFirst = ritorna il valore del primo elemento inserito nella queue.
- setFirst = modifica il valore del primo elemento inserito nella queue. Nel caso lo si utilizzi su una queue vuota restituisce un messaggio di errore.
- getLast = ritorna il valore dell'ultimo elemento inserito nella queue.
- setLast = modifica il valore dell'ultimo elemento inserito nella queue. Nel caso lo si utilizzi su una queue vuota restituisce un messaggio di errore.

La queue utilizza un iteratore di tipo Forward, sia in lettura che in scrittura. Il valore di begin dell'iteratore si inizializza con il primo valore inserito nella queue, mentre il valore di end è nullo.

Il metodo templato globale transformif modifica la coda con un determinato operatore di tipo F se, dato un determinato predicato di tipo P, esso viene verificato.

MAIN

Nel main richiamo delle funzioni di test che usano ogni metodo pubblico della queue, testando la coda con vari tipi di dati: interi, stringhe, struct "point" e liste di interi.
In esse viene definito il funtore di uguaglianza.

SCELTE IMPLEMENTATIVE

- La scelta dell'utilizzo di due puntatori serve per avere le operazioni di inserimento e rimozioni eseguibili in tempo costante.
- Ho preferito aggiungere nella file.h un metodo per la stampa della coda anche se non richiesto per avere, durante la fase di testing, maggiore chiarezza.
- Per la gestione del caso in cui si eseguono delle operazioni specifiche su queue vuota, ho preferito visualizzare un messaggio di errore per qlle eventualità.
- Ho utilizzato gli iteratori Forward essendo più che sufficienti per questo progetto.