Filippo Tibiletti mat. 887914

C++

Il file main.cpp comprende una serie di test per verificare il correttamento funzionamento della classe Queue.hpp.

- void testInteri( ):
  Verifica il corretto funzionamento della classe che crea la coda quando lavora con interi.

- void testStringhe( ):
  Verifica il corretto funzionamento della classe che crea la coda quando lavora con stringhe.

- void testStructVeicolo( ):
  Verifica il corretto funzionamento della classe che crea la coda quando lavora con la struttura Veicolo.

- void testCostruttori( ):
  Verifica che i costruttori e metodi per copiare funzionino correttamente.

- void testIteratori( ):
  Verifica che gli iteratori usati per navigare la classe Queue lavorino correttamente

Il file Queue.hpp implementa una coda generica con ordinamento FIFO

La coda viene costruita tramite dei nodi, definiti da una struct che contiene il valore effettivo ed un puntatore al nodo successivo.
È presente un costruttore che assegna il parametro ricevuto come valore del nodo

La classe Queue presenta altri attributi chiamati front, back e size.
front e back sono puntatori di tipo Node che puntano rispettivamente al primo e ultimo elemento della coda.
Size è un contatore usato per tenere traccia del numero di elementi presenti nella coda

Metodi Privati

La classe presenta una funzione friend chiamata transformif( ).
Viene messa come friend per garantire accesso anche ai dati privati

Metodi Pubblici

- Queue( ):
  Costruttore default della classe, imposta front e back a nullptr e size a 0

- Queue(const Queue& other):
  Copy constructor, prende come parametro un'altra coda e ne restituisce una copia

- Queue& operator=(const Queue& other):
  Controlla se la coda passata come parametro e la coda che invoca il metodo sono diverse, se vero crea una copia della coda parametro e la restituisce

- ~Queue( ):
  Distruttore della classe, richiama il metodo clearQueue( ) per svuotare correttamente la coda

- void enqueue(const T& value):
  Metodo usato per la creazione di un nuovo nodo.
  Inizializza un nuovo nodo, li assegna come valore il parametro ricevuto e alla fine aggiunge il nodo alla coda.
  Se la coda non presenta nodi inizializza testa e coda

- void enqueue(const_iterator begin, const_iterator end):
  Prende come parametri due iteratori, uno di inizio e uno di fine.
  Tramite gli iteratori legge i nodi da una coda e tramite il metodo enqueue(const T& value) li inserisce nella coda chiamate

- void dequeue( ):
  Metodo usato per rimuovere il valore in testa alla coda

- T& frontElement( ):
  Metodo usato per accedere in lettura e scrittura al primo elemento della coda.

- const T& forntElement( ):
  Metodo usato per accedere solamente in lettura al primo elemento della coda.

- T& backElement( ):
  Metodo usato per accedere in lettura e scrittura all'ultimo elemento della coda.

- const T& backElement( ):
  Metodo usato per accedere solamente in lettura all'ultimo elemento della coda.

- bool contains(const T& value):
  Metodo usato per verificare se il parametro ricevuto corrisponde ad almeno uno dei valori dei nodi della coda.
  Se presente il metodo restituisce true.

- size_t getSize( ):
  Metodo usato per restituire il numero di elementi presenti nella coda.

- bool isEmpty( ):
  Metodo booleano, restituisce true se la coda è vuota

- void printQueue( ):
  Metodo che tramite iteratori legge tutti nodi della coda e ne stampa i valori.

- const_iterator begin( ):
  Metodo usato per assegnare il valore corretto agli iteratori che puntano al primo elemento della coda

- const_iterator end( ):
  Metodo usato per assegnare il valore corretto agli iteratori che puntano all'ultimo elemento della coda

- void clearQueue( ):
  Metodo che cicla fino a quando la coda non risulta vuota, ad ogni iterazione chiama il metodo dequeue( ) per rimuovere correttamente il primo nodo dalla coda

È presente anche una funzione globale chiamata
tranformif(Queue<Q>& q, P p, F f).

Questa classe è templata allo stesso tipo della coda e presenta tre parametri:

- q: coda da processare
- p: predicato per verificare se modificare un elemento della coda
- f: funtore da applicare agli elementi da modificare

QT

Progetto con lo scopo di creare un editor di testi.
Il programma deve permettere di aprire, salvare e modificare un file .txt o .md

Le funzioni di salvataggio sono ‘Salva’ e ‘Salva con Nome’
Le modifiche avvengono tramite una dialog attivato da un pulsante ‘Cerca’ e si dividono in ‘Trova’, ‘Sostituisci’ e ‘Sostituisci Tutto’

mainwindow.h e mainwindow.cpp

File che vengono usati per la creazione della schermata principale del programma.

Presentano uno spazio vuoto dove permettere la visualizzazione e modifica di file testuali

Nella parte superiore è presente un bottone chiamato ‘File’ che se premuto apre un menu con altri bottoni:

Nuovo: Permette la creazione di un nuovo file
Salva: Permette il salvataggio del file attualmente in uso
Salva con nome: Permette di salvare il file specificando nome ed estensione
Cerca: Apre una nuova dialog per la ricerca e modifica

Metodi:

- void openFile( ):
  Apre una finestra per la selezione del file da caricare

- void saveFile( ):
  Metodo usato per salvare il file attualmente aperto

- void saveFileAs( ):
  Metodo usato per salvare il file attualmente aperto con un nuovo nome
  Apre una finestra per selezione percorso e assegnare nome ed estensione

- void openFindReplaceDialog( ):
  Metodo usato per l'apertura del menu di modifica

findreplacedialog.h e findreplacedialog.cpp

File usati per la definizione delle funzioni e dialog di ricerca e sostituzione di testo

I file definiscono campi, checkbox, bottoni e metodi

textEdit: Campo per il testo
findLineEdit: Campo per la stringa da cercare
replaceLineEdit: Campo per la stringa con cui sostituire
findButton: Bottone per la ricerca
replaceButton: Bottone per la sostituzione
replaceAllButton: Bottone per sostituire tutte le occorrenze

caseSensitiveCheckBox: CheckBox per abilitare il case sensitive
wholeWordCheckBox: CheckBox per abilitare la ricerca della parola completa

Metodi:

- void findText( ):
  Funzione che cerca e evidenza nel testo la stringa di ricerca inserita. Permette di usare dei flag per una ricerca migliore

- void replaceText( ):
  Funziona che si occupa di sostituire il testo trovato dalla findText( ) con la stringa ricevuta nel campo findLineEdit

- void replaceAllText( ):
  Funzione che sfrutta findText( ) e replaceText( ) in modo ciclico per trovare e sostituire tutte le occorrenze di stringhe inserite dall'utente