Alessandro Guerrisi MATR. 890331 a.guerrisi2@campus.unimib.it

RELAZIONE PROGETTO C++ E QT

PARTE 1: Progetto C++

Introduzione

Il progetto richiede la progettazione e realizzazione di una classe che implementa un Set di elementi generici T. Un Set è una collezione di dati che NON può contenere duplicati: es. un set di interi S = {1,6,4,9,7,10,12}.

Per l'implementazione NON POTETE USARE STRUTTURE A LISTA. A parte i metodi essenziali per il suo corretto uso, la classe deve implementare anche le seguenti funzionalità:

1. Deve esistere un metodo add per l'aggiunta di un elemento.
2. Deve esistere un metodo remove per la rimozione di un elemento.
3. Accesso, in sola lettura, all'i-esimo elemento tramite operator[ ]
4. Un metodo contains, che ritorna true se il set contiene un dato valore passato come parametro.
5. La classe deve essere dotata solo di const_iterator.
6. Deve essere possibile creare un Set a partire da una sequenza di dati definita da una coppia generica di iteratori su tipi Q. Lasciate al compilatore la gestione della compatibilità tra i tipi attraverso l'uso del cast.
7. Deve essere possibile stampare il contenuto del set utilizzando operator<<. Il formato di stampa deve essere il seguente: stampare la dimensione del set e sulla stessa riga, separati da spazi, i valori contenuti nel set (formattati liberamente) tra parentesi tonde. Ad esempio, per il set S = {1, 6, 4, 9, 7, 10, 12}, la stampa deve produrre la stringa: "7 (1) (6) (4) (9) (7) (10) (12)".
8. Implementare un metodo, tramite operator==, per confrontare due set e ritorna true se i due set contengono gli stessi dati.

Implementare una funzione globale e generica filter_out che, dato un Set generico S su tipi T e un predicato booleano generico P, ritorna un nuovo set di tipi T ottenuto prendendo da S tutti gli elementi che soddisfano il predicato P.

Implementare una funzione globale operator+ che, dati in input due Set generici su tipi T, ritorna un Set di tipi T che contiene gli elementi di entrambi i set ("concatenazione" di set).

Implementare una funzione globale operator- che, dati in input due Set generici su tipi T, ritorna un Set di tipi T che contiene gli elementi comuni a entrambi i set ("intersezione" di set).

Implementare una funzione globale save che dato in input un set che contiene dati di tipo std::string e un nome di file, salva il contenuto del set in un file testuale.

Alessandro Guerrisi MATR. 890331 a.guerrisi2@campus.unimib.it

Scelte progettuali

Il progetto contiene due file principali:

- File set.hpp: Rappresenta il file header della mia classe Set templata. All'interno viene dichiarata la struttura di un Set di elementi di tipo generico insieme ai suoi metodi. La classe Set è una classe templata con due valori, un tipo T generico perché la struttura deve contenere diversi tipi di dati (primitivi e custom) e un funtore Equal che verrà definito nel file main.cpp per stabilire l'uguaglianza tra due dati di tipo T. La classe ha due attributi fondamentali che sono "size", che rappresenta la grandezza del Set, e "elements" ovvero un puntatore a dati di tipo T.
  Quando viene creato un Set viene assegnato al puntatore un array di elementi di lunghezza 0 e impostata la grandezza della struttura a 0. Viene utilizzato solo il costruttore senza parametri per garantire il miglior funzionamento nel Set nel caso dell'aggiunta o della rimozione di un elemento dal Set.
  Considerando il fatto che un Set non può contenere elementi duplicati la funzione "contains()" è fondamentale per l'aggiunta o la rimozione di elementi. Questa viene trovata nei metodi chiamati "add()" e "remove()". Per prevenire memory leaks viene usato in entrambi i metodi un array di supporto chiamato "new_elements" nel quale verranno inseriti gli elementi del nostro attuale Set per poi essere ritrasferiti in maniera corretta con l'elemento aggiunto o rimosso. Infine, l'array di supporto viene deallocato correttamente dalla memoria.
  Il metodo "remove()" gestisce sia il caso in cui l'elemento da rimuovere sia in ultima posizione sia il caso in cui sia in mezzo all'array, in questo caso tutti gli elementi successivi verranno scalati per non lasciare buchi.
  Nella classe vengono anche implementati dei const_iterator di categoria bi-direzionale, questo perché , non essendo una struttura a lista nel quale si può puntare solo all'elemento successivo, in questa classe viene implementato anche l'operator-- per poter accedere anche agli elementi precedenti a quello corrente.
  Il Set contiene anche quattro funzioni globali:
  - Una funzione chiamata "filter_out()" che tramite l'utilizzo di iteratori scorre il Set e stampa solo gli elementi che soddisfano un predicato generico di tipo P che verrà definito nel file main.cpp
  - Una funzione di concatenazione di elementi chiamata "operator+" e una funzione di intersezione di elementi chiamata "operator-". Entrambe prendono come argomenti due Set di stesso tipo e grazie al metodo "contains()" controllano se è presente o meno l'elemento in entrambi i Set. Viene usato "tmp" che è un'ulteriore Set di supporto che conterrà la concatenazione o l'intersezione delle due strutture.
  - Una funzione chiamata "save()" che ha come argomento un Set di dati e un file dove verrà stampato il contenuto della struttura. Gli elementi saranno limitati da parentesi tonde e divisi tra loro con una spaziatura. Il metodo gestisce anche l'eccezione nel caso il file non venga aperto correttamente. Se il file non esiste viene creato, mentre se è già presente viene sovrascritto il contenuto.

- File main.cpp: Rappresenta il file di test relativo alla struttura dati Set. In questo file sono presenti diversi metodi che testano le varie funzionalità della struttura come l'aggiunta o la rimozione di elementi, ma anche test per l'utilizzo degli iteratori per scorrere lungo il Set.
Il file è diviso in diverse sezioni:
  - I funtori: sono presenti funtori utili per la creazione di diversi tipi di Set come "equal_int" che valuta l'uguaglianza tra interi e che serve per la creazione di un Set contenente valori interi, oppure funtori utili per funzioni globali della classe Set come "is_even" che verrà usato come predicato per la funzione "filter_out()".
  - Test per tipi primitivi: dove vengono testati i Set contenenti elementi di tipo primitivo come interi e stringhe
  - Test per tipi custom: dove vengono testati i Set contenenti elementi di tipo custom. Viene definito un tipo custom chiamato "person" che ha un nome di tipo string e un'età di tipo int. Per questo viene definito un ulteriore funtore per l'uguaglianza tra due persone chiamato "equal_person" e inoltre viene sovrascritto l'operatore di stampa "operator<<" per poter stampare correttamente un Set contenente elementi di tipo "person".
  - Test per iteratori: dove vengono testati i diversi metodi degli iteratori implementati dalla classe Set, come la stampa, l'incremento, l'uguaglianza, ecc.
  - Test funzioni globali: dove vengono testate le funzioni globali della classe Set. Nel test della funzione globale "filter_out()" viene inoltre testato il punto 6. della richiesta del progetto.

PARTE 2: Progetto Qt

Introduzione

Il progetto richiede la progettazione e realizzazione di un'applicazione con GUI per la gestione dei dipinti presenti nella Galleria degli Uffizi. Per la gestione degli elementi della collezione servirsi della classe Set implementata per il progetto precedente. L'applicazione dovrà caricare le informazioni contenute nel file "dipinti_uffizi.csv" in una tabella i cui valori riportati nelle celle NON devono poter essere modificati.

Dovranno inoltre essere implementate le seguenti funzionalità:

1. Un pulsante per l'aggiunta di un nuovo dipinto;
2. Un pulsante per la rimozione di un dipinto;
3. Uno strumento di ricerca di un dipinto in base al campo Soggetto/Titolo;
4. Un grafico che riporti la percentuale di dipinti per ciascuna Scuola ed un altro grafico in cui venga riportato il numero di dipinti rispetto al campo Data.

Scelte progettuali

Il progetto presenta un'applicazione con GUI gestita in questo modo:

Vengono prelevati dal file "dipinti_uffizi.csv" diversi dipinti presenti nella Galleria degli Uffizi in base a 5 campi diversi che in ordine sono "Scuola", "Autore", "Soggetto/Titolo", "Data" e "Sala". Questi dati vengono prelevati grazie alla funzione "loadDataCsv()" che si occuperà di aggiungere

correttamente i dipinti in una QTableWidget. I dipinti verranno aggiunti anche in un Set di dati chiamato "paintings" implementato includendo il file "set.hpp" della parte precedente del progetto. Viene aggiunta una struct apposita per i dipinti chiamata "painting" con anche il suo funtore "equal_painting" per verificare l'uguaglianza (due dipinti con lo stesso nome sono uguali). Per poter aggiungere correttamente le linee del file csv la funzione "loadDataCsv()" si appoggia alla funzione "parseCSVLine()" che parsa correttamente ogni linea del file. I campi vengono divisi da virgole a meno che non viene trovato un doppio apice e a quel punto il campo va fino a quando viene trovato l'altro doppio apice di chiusura.

L'applicazione quindi presenta una tabella contente tutti i dipinti correttamente e una serie di pulsanti per modificare i valori di questa:

- Pulsante "Aggiungi": Pulsante che permette di aggiungere un nuovo dipinto. Il pulsante diventa selezionabile solamente quando tutti i campi del dipinto sono non vuoti. Questo perché non è possibile aggiungere un dipinto a cui mancano delle informazioni. Prima di aggiungere un nuovo dipinto viene verificato se questo dipinto non è già presente nel Set di dipinti chiamato "paintings", in caso contrario non saranno presenti modifiche.
- Pulsante "Rimuovi": Pulsante che permette di rimuovere un dipinto presente nella tabella tramite il suo titolo. Verrà cercato il dipinto con tale titolo e verrà eliminato sia dalla tabella che anche dal Set di dipinti chiamato "paintings" se questo è presente nel Set di dipinti chiamato "paintings", in caso contrario non saranno presenti modifiche.
- Pulsante "Cerca": Pulsante che permette di cercare un dipinto presente nella tabella tramite il suo titolo. Verrà cercato il dipinto con tale titolo, se viene trovato allora la tabella si sposterà sul dipinto e lo selezionerà.

I metodi che gestiscono il click di questi 3 pulsanti chiamano un metodo "updateUI()" che aggiornerà l'interfaccia facendo tornare vuoti i campi per l'aggiunta, la rimozione, la ricerca e inoltre farà tornare non selezionabile il pulsante "Aggiungi".

Presenti anche due ulteriori pulsanti per generare i grafici:

- Pulsante "Genera grafico a torta": Pulsante che genera un grafico a torta ricevendo i dati dalla tabella contenente i dipinti. Mostra le percentuali di dipinti, rispetto al totale, per ogni scuola. Per salvare ogni scuola con il relativo numero di dipinti viene usata una QMap e il numero di dipinti viene poi convertito in percentuale.
- Pulsante "Genera grafico a barre": Pulsante che genera un grafico a barre ricevendo i dati dalla tabella contenente i dipinti. Mostra il numero di dipinti per ogni data presente. Per salvare ogni data con il relativo numero di dipinti viene usata una QMap.

Il metodo "toLower()" presente in diverse porzioni di codice serve per convertire una QString in minuscolo. Questo per cosentire l'aggiunta, la modifica o la ricerca di un dipinto senza preoccuparsi di rispettare le maiuscole e le minuscole. Ad esempio, se si vuole cercare il dipinto con titolo "Apollo e Marsia", se nel campo per la ricerca verrà inserito "apollo e marsia" il dipinto verrà trovato ugualmente.