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

RELAZIONE PROGETTO C++ E QT

PARTE 1: Progetto C++

Introduzione

Il progetto richiede la progettazione e realizzazione di una classe generica che implementa un grafo orientato. Un grafo è costituito da un insieme di nodi e archi. I nodi sono rappresentati da un generico identificativo T (es. un numero, una stringa, un oggetto, ecc...). Gli archi mettono in relazione due nodi creando un collegamento tra loro.

Requisito essenziale del progetto è che il grafo deve essere implementato mediante matrici di adiacenza di booleani come in figura. Non possono essere usate liste. A parte i metodi essenziali per la classe (tra cui conoscere il numero di nodi e archi), devono essere implementate le seguenti funzionalità:

1. La classe deve includere il supporto al solo const_iterator di tipo forward sui nodi. L'iteratore itera sull'insieme degli identificativi dei nodi contenuti nel grafo e l'ordine con cui vengono ritornati non è rilevante.
2. Deve essere possibile aggiungere e rimuovere un nodo usando il suo identificativo. Quando si aggiunge o si rimuove un nodo, bisogna aggiornare la matrice di adiacenza. Gestire i casi: nodo da aggiungere già esistente, nodo da eliminare non esistente.
3. Deve essere possibile aggiungere e rimuovere un arco usando la coppia di identificativi dei nodi che collega. Quando si aggiunge o si rimuove un arco, bisogna aggiornare la matrice di adiacenza. Gestire i casi: arco non esistente e arco già esistente.
4. Deve essere possibile interrogare il grafo per sapere se esiste un nodo tramite un metodo "existsNode()" e se una coppia di nodi è connessa da un arco tramite un metodo "existsEdge()". Ad esempio se in un grafo non esiste il nodo 9 allora existsNode(9)=false. Mentre se esistono il nodo 1 e il nodo 2 con un arco che parte da 1 e arriva a 2 allora existsEdge(1,2)=true. L'arco (2,1) è diverso dall'arco (1,2).

Scelte progettuali

Il file contiene due file principali:

- File "graph.hpp": Rappresenta il file header della mia classe graph templata. All'interno viene dichiarata la struttura di un grafo orientato insieme ai suoi metodi per la modifica e l'aggiornamento. La classe graph è una classe templata con due valori, un tipo T generico perché la struttura deve poter contenere nodi rappresentati da un generico identificativo, e un funtore Equal che verrà definito nel main.cpp per stabilire l'uguaglianza tra due nodi con identificativo di tipo T. La classe ha 4 attributi fondamentali cioè "n_nodes", che rappresenta il numero di nodi del grafo, "n_edges", che rappresenta il numero di archi nel grafo, "nodes", che è un puntatore ai dati di tipo T e infine "matrix" che è puntatore rappresentante la matrice di adiacenza di booleani.

Quando viene creato un grafo vengono inizializzati il numero di nodi e di archi a 0 e i due puntatori a nullptr.
E' possibile aggiungere un nodo tramite il metodo "addNode()" (se il nodo passato come parametro non è già presente) e rimuovere un nodo tramite il metodo "removeNode()" (se il nodo passato come parametro è presente nel grafo). Se si prova ad aggiungere un nodo già presente o a rimuovere un nodo non presente verranno lanciate le relative eccezioni.
E' possibile aggiungere un arco tramite il metodo "addEdge()" (se non esiste già un arco che collega i 2 nodi passati come parametri) e rimuovere un arco tramite il metodo "removeNode()" (se esiste un arco che collega i 2 nodi passati come parametri). Se si prova ad aggiungere un arco già presente o a rimuovere un arco non presente verranno lanciate le relative eccezioni.
Per controllare se è presente o meno un nodo o un arco verranno utilizzate le funzioni "existsNode()" e "existsEdge()".
Un altro metodo fondamentale è il metodo "indexNode()" che restituisce l'indice del nodo passato come parametro. Questo metodo è molto utile perché semplifica molto la gestione del programma in quanto permette ad esempio di trovare direttamente le coordinate della matrice di adiacenza e di aggiornarla correttamente.
In entrambi i metodi per l'aggiunta e la rimozione vengono usati due puntatori di supporto chiamati "tmp", nel quale salveremo i nodi attuali del grafo, e "matrix_tmp", nel quale salveremo l'attuale matrice di adiacenza. In questo modo il grafo e la matrice di adiacenza vengono gestiti in maniera dinamica consentendo l'aggiunta o la rimozione di nodi senza causare problemi di memoria. Nel caso in cui l'assegnamento dei valori attuali all'interno dei puntatori di supporto non dovesse andare a buon fine verranno deallocati correttamente dalla memoria. Al termine dei due metodi il grafo e la matrice saranno aggiornati correttamente con l'elemento aggiunto o rimosso.
Nella classe vengono anche implementati dei const_iterator di tipo forward per permettere l'iterazione sull'insieme degli identificativi dei nodi contenuti nel grafo. Il loro uso viene testato nel file main.cpp.
Nella classe è anche presente un metodo "empty()" che si occupa di deallocare correttamente le risorse al termine del programma.

- File "main.cpp": Rappresenta il file di test relativo alla classe che rappresenta il grafo orientato. In questo file sono presenti diversi metodi che testano le varie funzionalità della struttura come l'aggiunta o la rimozione di nodi e archi, le interrogazioni sul grafo come la presenza di archi e nodi, ma anche test per l'utilizzo degli iteratori per scorrere lungo i nodi del grafo. Il file è diviso in diverse sezioni:

  - Test per metodi fondamentali: dove vengono testati i metodi fondamentali della classe graph. Inizialmente viene creato un grafo di nodi con identificativo intero chiamato "graphI" e nel quale vengono aggiunti 3 nodi e un arco. In seguito vengono creati 2 grafi usando il Copy Constructor e l'operatore di assegnamento. Per entrambi i grafi viene testata la loro struttura per vedere se entrambi sono stati creati correttamente e con gli stessi dati del grafo di partenza.

- Test per grafi di interi: dove vengono testati tutti i metodi di gestione di un grafo di interi come l'aggiunta, la rimozione e l'interrogazione. Gestiti anche i casi in cui viene lanciata un'eccezione come, ad esempio, nell'aggiunta di un nodo già presente.

- Test per grafi di stringhe: dove vengono testati tutti i metodi di gestione di un grafo di stringhe come l'aggiunta, la rimozione e l'interrogazione. Gestiti anche i casi in cui viene lanciata un'eccezione come, ad esempio, nell'aggiunta di un nodo già presente.

- Test per tipi custom: dove vengono testati tutti i metodi di gestione di un grafo contente nodi con identificativo 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".

- Test per iteratori: dove vengono testati i diversi metodi degli iteratori implementati dalla classe graph, come la stampa, l'incremento, l'uguaglianza, ecc.

PARTE 2: Progetto QT

Introduzione

Il progetto d'esame richiede la progettazione e realizzazione di un'interfaccia grafica per la visualizzazione di un Pokédex.

L'interfaccia sarà costituita da una finestra principale, in cui visualizzare l'elenco dei Pokèmon con le informazioni relative a ciascuno di essi e una seconda finestra in cui è possibile confrontare le caratteristiche tra due Pokémon distinti. Per l'elenco dovranno essere implementate le seguenti funzionalità:

1. Visualizzazione delle informazioni per tutti i 1190 Pokémon (contenute nel file "pokedex.csv"). Per ciascuna riga riportare le informazioni relative a ciascuno (nome, tipo, attacco, ecc.);
2. Inclusione/esclusione dall'elenco in base alla tipologia di Pokémon selezionata/deselezionata;
3. Ordinamento dell'elenco in base a una delle colonne della tabella.

Per la finestra di confronto sono richieste le seguenti funzionalità:

1. Selezione dei due Pokémon da confrontare (non deve essere consentito confrontare un Pokémon con sé stesso);
2. Visualizzazione dell'immagine raffigurante ciascun Pokémon (immagini presenti nella cartella "images") e della tipologia di Pokémon.
3. Creazione e visualizzazione di un grafico in cui riportare il confronto tra i due Pokémon.

Scelte progettuali

L'interfaccia grafica progettata è costituita da due diverse finestre. La prima è chiamata "MainWindow" e il suo compito è quello di far visualizzare a schermo una tabella contenente una lista di Pokèmon e di fornire una serie di servizi come l'inclusione/esclusione di Pokèmon o l'ordinamento rispetto a una delle colonne della tabella. La seconda finestra chiamata

"ConfrontWindow" invece consente di confrontare due distinti Pokèmon selezionati e di mostrare tramite un grafico le differenze tra le statistiche. In seguito, vengono elencate le varie funzionalità e le scelte progettuali:

- MAINWINDOW: Alla creazione di questa finestra vengono prelevati dei dati da ogni riga del file "pokedex.csv" e salvati in una struttura chiamata "Pokemon". Questa struttura rappresenta un Pokèmon con le proprie statistiche come "img", "Name", "Type", "Attack", "Defense", ecc. Quando il prelievo dei dati è concluso vengono quindi salvati tutti questi Pokèmon all'interno di una QList chiamata "listaPokemon" che rappresenta la lista completa. Questa verrà passata al metodo "populateTable()" per popolare inizialmente una TableWidget. Oltre a questa tabella saranno presenti diversi pulsanti:

  - Pulsante "Filtra tabella": pulsante che consente di filtrare la lista di Pokèmon per tipologia. Essendo che un Pokèmon può avere più tipologie l'utente può inserire più tipologie da filtrare. Ad esempio:
    - Viene digitato "Fire", la tabella mostrerà tutti i Pokèmon che contengono la tipologia "Fire", quindi sia i Pokèmon con singolo tipo "Fire" sia quelli con doppio tipo come "Ghost" "Fire".
    - Viene digitato "Fire" e "Flying", la tabella mostrerà solo i Pokèmon con entrambe le tipologie.

  - Pulsante "Resetta tabella": pulsante che consente di resettare la tabella con la lista completa di Pokèmon senza dover necessariamente chiudere e riaprire la Window.

  - Pulsante "Attiva Funzione di ordinamento": pulsante che attiva la funzione che rende possibile l'ordinamento in base a una delle colonne della tabella.

  - Pulsante "Disattiva Funzione di ordinamento": pulsante che disattiva la funzione di ordinamento. Inizialmente la tabella viene impostata con la funzione di ordinamento disabilitata.

  - Pulsante "Apri Confronto": pulsante per l'apertura della ConfrontWindow nella quale si potranno confrontare le statistiche tra due Pokèmon.

I pulsanti "Resetta tabella" e "Filtra tabella" inoltre sono fatti in modo tale che quando vengono cliccati viene disattivata la funzione di ordinamento. Questo per garantire la corretta visualizzazione della lista completa o filtrata. Una volta resettata o filtrata la lista è possibile riattivare la funzione di ordinamento tramite il pulsante "Attiva funzione di ordinamento".

- CONFRONTWINDOW: All'interno di questa finestra sono presenti due "ComboBox", ovvero dei menù a tendina, per la selezione di Pokèmon dalla lista completa ottenuta dalla MainWindow. Una volta selezionato un determinato Pokèmon verranno mostrate a schermo la sua immagine e la sua tipologia, sia nel caso in cui abbia singolo tipo e sia nel caso in cui abbia doppio tipo. Se i due Pokèmon selezionati dai menù saranno differenti allora il pulsante "Confronta" diventerà attivo e al suo click verrà creato il grafico di confronto. In questo grafico a barre verticali saranno quindi mostrate le differenze tra le statistiche dei due Pokèmon.