Relazione progetto C++ Febbraio 2022

Nome: Matteo

Cognome: Covelli

Matricola: 861277

Mail: m.covelli7@campus.unimib.it

INTRODUZIONE

Dopo un'accurata valutazione del problema ho deciso di implementare la classe generica Set utilizzando le linked list.

La lista è costituita da nodi, dove il nodo è composto da un puntatore al nodo successivo e da un valore generico.

TIPI DI DATI

Per garantire la corretta costruzione della set e quindi implementare la lista, ho utilizzato una struttura dati (struct): nodo.

Il nodo è la struttura che contiene il puntatore al nodo successivo e il valore generico.

IMPLEMENTAZIONE – METODI IMPLEMENTATI

La classe set è costituita dai seguenti attributi: il puntatore al nodo head (primo nodo della collezione di elementi) e il funtore equals per quanto riguarda l'uguaglianza tra dati generici.

All'interno della classe set ho implementato i 4 metodi fondamentali, ovvero il costruttore, l'operatore di assegnamento, il copy constructor e il distruttore. Questi ultimi due metodi utilizzano una funzione chiamata svuota.

Successivamente basandomi sulle richieste della traccia di esame ho deciso di implementare diversi metodi:

Il primo è il **costruttore secondario** per set, che garantisce la creazione della Set a partire da una sequenza di dati definita da una coppia generica di iteratori su tipi Q.

Il secondo è il metodo **add**, il quale permette di effettuare l'aggiunta di un valore all'interno della set.

Il terzo è il metodo **remove**, il quale permette di rimuovere un determinato valore all'interno della set.

Il quarto è il metodo **operator[]**, che permette l'accesso, in sola lettura, all'i-esimo elemento della set.

Il quinto è il metodo **operator==**, che permette di verificare se due set contengono gli stessi dati, confrontandole.

Il sesto metodo è la ridefinizione dell'operatore di stream<< per la set (**operator<<**). Come afferma il nome, permette la stampa in output della set.

Il settimo metodo che ho implementato è **svuota**, il quale viene utilizzato all'interno del distruttore, copy constructor, costruttore secondario della set e nelle tre funzioni globali. Esso permette di svuotare la lista eliminando tutti i nodi al fine di garantire una corretta gestione della memoria.

Ho usato anche tre metodi di supporto: **trovato**, **sizeCollezione** e **vuota**. Il primo mi permette di verificare se un valore è presente all'interno della set. Il secondo mi restituisce la grandezza della set. Infine il terzo mi restituisce true/false se una set è vuota.

Inoltre la classe set implementa un **const iterator** di tipo forward.

Infine ho implementato 3 funzioni globali:

La prima funzione globale generica è **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.

La seconda funzione globale è **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).

L'ultima funzione globale è **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).

MAIN

Ho utilizzato il main per effettuare tutti i test del caso sulla set. In particolar modo ho effettuato i test su set utilizzando tipi custom e non.

Il primo test che ho effettuato è stato su una set di interi, nel quale ho testato tutti i metodi implementati. Successivamente ho effettuato i medesimi test su una set di stringhe.

Un ulteriore test che ho svolto è stato su una set di interi passata costante come parametro.

Un altro test fondamentale che ho svolto è stato su una set di tipo custom, composta da punti. Per effettuare questo test ho implementato nel main, la struct punto. Anche qui ho testato tutti i metodi fondamentali.

Un altro test che ho implementato è quello sulla filter_out, nel quale ho usato una set di punti. Per testare questo metodo ho definito tre metodi nel main (cordinatePari, cordinateDispari, cordinateUguali).

L'ultimo test che ho effettuato è quello sull'iterator, testando ogni metodo di quest'ultimo sia con set di tipo intero e set di tipo custom (punto).

Inoltre nel main ho aggiunto tre struct per quanto riguarda l'equals; in particolare equalsInt, equalsString e equalsPunto, fondamentali per poter fare confronti tra set con i diversi tipi di valori.