RELAZIONE PROGETTO C++ 2022

1 SETTEMBRE – 17 SETTEMBRE

Programmazione C++
Autore: Davide Collato
Matricola: 858025
Email: d.collato@campus.unimib.it

Introduzione

Il progetto richiedeva l'implementazione di un multiset ordinato di dati generici T. Siccome il multiset può contenere duplicati ma, per minimizzare la gestione della memoria, esso è stato implementato come un array di struct element.

Element

Rappresenta l'elemento contenuto nel multiset. Possiede come dati membro un valore di tipo T, value, che rappresenta il valore contenuto nel multiset, e un unsigned int, ocnumber, che rappresenta il numero di occorrenze.
Oltre ai metodi fondamentali, sono stati implementati anche due costruttori secondari, uno che prende come parametro un valore con cui inizializzare value ed uno che prende come parametri sia un valore con cui inizializzare value sia uno con cui inizializzare ocnumber.

Multiset

Il multiset possiede 4 dati membro: un puntatore a element (_head) che rappresenta il mio array di elementi, un unsigned int (_size) che rappresenta la lunghezza dell'array (ossia il numero di elementi senza ripetizioni) e due funtori, rispettivamente il funtore di uguaglianza (_equals) e quello di confronto (_compare). Sono stati implementati i metodi fondamentali che qui brevemente elenco:

- Costruttore di default che inizializza _head a nullptr e _size a 0,
- Distruttore che tramite il metodo clear svuota l'array e porta la classe in uno stato coerente,
- Copy constructor (si rimanda alla documentazione),
- Operatore di uguaglianza (si rimanda alla documentazione).

Metodi implementati

In ordine di scrittura, sono stati implementati i seguenti metodi:
Metodo **find** (privato) che, preso in input un valore, restituisce il puntatore all'elemento che lo contiene, altrimenti restituisce nullptr.
Successivamente abbiamo il metodo **swap** il quale, preso come parametro un multiset (chiamato other) scambia i dati membro di this con quelli di other (cioè _head e _size).
Un metodo **tot_elementi** che restituisce il numero totale di valori contenuti nel multiset, contando il numero di occorrenze e un metodo **occurencies** che accetta in ingresso un dato di tipo T e ritorna il numero di occorrenze di quel dato.
Successivamente ho implementato il metodo **add** in due varianti: una che prende in input un valore (di tipo T) e una che prende come parametro un valore e un numero di occorrenze (unsigned int). L'aggiunta del valore al multiset avviene seguendo la policy di ordinamento implementata dall'utente. L'aggiunta avviene attraverso la creazione di un multiset temporaneo di _size = _size + 1 su cui poi eseguiamo la swap degli elementi con this. È stato implementato anche un metodo **remove** che, preso come parametro un valore, lo rimuove dal multiset. Se il valore non è presente nel multiset, viene lanciata un'eccezione custom. La rimozione avviene attraverso la creazione di un multiset temporaneo di _size = _size - 1 su cui poi eseguiamo la swap degli elementi con this.
Ho implementato inoltre un metodo **contains** che ritorna true se il valore di input è contenuto nel multiset (fa uso del metodo find), l'**operatore di uguaglianza** che ritorna true se due multiset contengono gli stessi elementi con lo stesso numero di occorrenze e l'**operatore di stream** (funzione globale) il quale stampa il multiset nella forma {<X1,occorrenzeX1>, …, <Xn, occorrenzeXn>}.
Per una descrizione più approfondita dei metodi, si rimanda alla documentazione.

Iteratori

È stato implementato un const_bidirectional_iterator perché, dato il fatto che la classe possiede un ordinamento, ho ritenuto potesse avere senso il poter scorrere il multiset in entrambi i sensi. Esso possiede come dati membro un puntatore a element e una variabile contatore di tipo unsigned int (utilizzata negli operatori di post e pre incremento e decremento). Inoltre è stato implementato un costruttore che, prendendo in input una coppia di iteratori, costruisce un multiset ordinato. È possibile stampare il multiset attraverso gli iteratori, ottenendo in questo modo la sequenza completa di tutti gli elementi contenuti nel multiset ordinato.

Main

Nel main sono state testate tutte le funzionalità dell'interfaccia pubblica su diversi tipi di dato: interi, stringhe, interi (imitando il gruppo Z_3), un dato custom point composta da una coppia di double, un dato custom contact composta da una terna di stringhe e multiset di multiset di interi.
L'esecuzione del test è accompagnata da varie stampe che descrivono all'utente le istruzioni che vengono compiute.

Altre informazioni

Per la scrittura del codice ho utilizzato Notepad++ v8.4.4 (32-bit), mentre per la compilazione è stato usato il sottosistema Linux per Windows con

- Ubuntu v20.04.3
- g++ v9.3.0
- valgrind v3.15.0