Progetto Set Febbraio 2022

Nome: [redacted]
Cognome: [redacted]
Matricola: [redacted]
Mail Ateneo: [redacted]

Introduzione

Il problema in questione ci chiedeva di rappresentare un Set. Un Set è una collezione di dati dello stesso tipo ma con tutti i valori diversi tra loro. Ho pensato a un Set come una linked-list, cioè elementi che non occupano memoria contigua nel computer ma piuttosto sono trovati tramite puntatori. La Set è ovviamente templata

Tipi di dati

Una Set, come anticipato prima, è quindi una collezione di strutture chiamate Element così definite:

- Value : è il valore di tipo T (valore generico che verrà poi stabilito a run-time in base al dato inserito dall'utente) da memorizzare all'interno del Set
- Next : è il puntatore di tipo element a un prossimo elemento del set

La Set possiede 3 attributi privati:

- M_head : è un puntatore di tipo elemento che punta all'elemento in testa al Set
- M_tail : è un puntatore di tipo element che punta all'ultimo elemento del Set
- M_size : è un attributo di tipo *unsigned int* che memorizza il numero di elementi presenti nel Set. È l'unico attributo che ha un metodo che funge da getter per poter essere ritornato il suo valore all'utente se richiesto.

Implementazione set

Come ogni classe che si rispetti (e che si vuole che funzioni bene) abbiamo bisogno dei quattro metodi fondamentali:

1. Costruttore di default

2. Distruttore
3. Copy constructor
4. Operator di assegnamento =

Il costruttore di default permette il settaggio dei parametri a un valore "non importa", precisamente la size della Set viene settata a 0 mentre il puntatore al primo elemento viene settato a *nulltptr* (in quanto non è presente alcun elemento all'interno della Set).

Il distruttore si occupa di eliminare, a fine programma, ogni allocazione di memoria usata durante l'esecuzione.

Il copy constructor permette di creare un nuovo Set uguale a quella passata come parametro (N.B un set si dice uguale se contiene gli stessi elementi, l'ordine degli elementi è ininfluente).

Operator= ha il compito di porre uguale due set. Al primo Set vengono inseriti gli elementi presenti all'interno del secondo set.

Metodi implementati

Facciamo una descrizione in ordine di visualizzazione dei metodi implementati all'interno del file "set.h"

Il metodo clear() ha il compito di svuotare un Set da tutti i suoi elementi ed imposta gli attributi della classe a un valore "non importa".

Il metodo add() ha il compito di inserire un elemento passato come parametro all'interno del Set. Questo metodo non inserisce un elemento se questo è già presente all'interno del Set in quanto non sono permessi duplicati. Ogni qual volta un metodo chiama la add questa è contenuta in un blocco try-catch in quanto potrebbe fallire. N.B. Dato che nella consegna non c'era alcun vincolo sul posizionamento degli elementi, l'elemento viene inserito in testa al Set.

Il metodo remove() ha il compito di togliere un elemento passato come parametro dal Set. Ha un caso base che consiste nel verificare se l'elemento in testa è quello da rimuovere. L'altro caso, tramite uno scorrimento di un pointer, individua e quindi procede all'eliminazione dell'elemento richiesto. Se l'elemento da rimuovere passato come parametro non viene trovato all'interno del Set il metodo si conclude. Il funzionamento è molto semplice, una volta individuato l'elemento da eliminare si fa scorrere un secondo po

Il metodo size() funge da getter per l'attributo della classe m_size.

Il metodo print() ha il compito di scorrere e quindi stampare gli elementi presenti nel Set

Come richiesto dalla consegna, si è implementato un costruttore che prendere come parametri due iteratori -> Set(Iterator, Iterator). Questi iteratori fanno parte di un

altro Set e hanno il compito di inserire gli elementi del secondo Set all'interno del primo. (in pratica un simil Copy Constructor)

Operator<<() è il metodo che ridefinisce lo stream. Tramite questa ridefinizione è possibile stampare il contenuto del Set con un normale std::cout senza chiamare il suo metodo print()

Operator==() è il metodo che ridefinisce l'operator che verifica l'uguaglianza. Si ha bisogno di questa ridefinizione in quanto l'operatore == di default non saprebbe in base a quale criterio rispondere se un set è uguale ad un altro o meno. Un set si dice uguale a un altro Set se contiene gli stessi elementi. N.B. la posizione degli elementi è ininfluente infatti il controllo di uguaglianza avviene solo sugli elementi e sulla size dei due Set

Operator[ ] è il metodo che ridefinisce l'operator parentesi quadra. La nuova funzione di questo operatore è quello di stampare a video l'elemento presente nell'indice richiesto. Se l'indice supera la size viene stampato un messaggio di errore.

Ci veniva inoltre chiesto di dotare la classe con *const_iterator* del quale abbiamo ridefinito due metodi

Begin() metodo che inizializza un iteratore in testa al Set

End() metodo che inizializza un iteratore in coda al Set

Funzioni globali

La consegna ci chiedeva di implementare tre funzioni globali templati che sono di seguito descritti

Filter_out() funzione che prende come parametri un Set di tipo T da filtrare, un predicato di tipo P e un valore di tipo T che fungono da filtro. Questo metodo ritorna un nuovo Set che contiene solo gli elementi del vecchio Set che rispettano i criteri dettati dalla coppia Predicato e valore filtro. Vengono istanziati due iteratori del Set da filtrare e per ogni valore trovato si verifica se questo elemento è ammissibile, e nel caso lo sia, lo si inserisce dentro un nuovo Set che sarà passato alla funzione chiamante.

Operator+() funzione che prende come parametri due Set e restituisce la loro concatenazione in nuovo Set. Il funzionamento è molto semplice, prende il primo ed inserisce i suoi valori, prende il secondo set ed inserisce i suoi valori. Il controllo se un elemento è già presente all'interno del nuovo Set avviene già alla chiamata del metodo add() quindi non c'è bisogno di scrivere ulteriore codice.

Operator()- funzione che prende come parametri due Set e restituisce la loro intersezione. Consiste in due cicli for annidati. Il primo for punta ad uno ad uno gli elementi del primo Set. Il secondo for ha il compito di scorrere il secondo Set per

trovare l'elemento attualmente puntato nel primo Set. Se questo elemento viene trovato in entrambi i Set allora lo si aggiunge a nuovo Set che poi verrà passato per valore alla funzione chiamante.

Funzione main

La funzione main si trova nel file "main.cpp" ed ha il compito di testare i vari metodi contenuti nel file "set.h"

Il main chiama sei funzioni, ognuno testa una parte del programma:

test_fondamenti_set_int() è il test dove si fanno eseguire i quattro metodi principali su un Set di tipo int

test_set_int() è il test dove si fanno eseguire i vari metodi implementati descritti precedentemente

test_set_int_filter() è il test che testa il filtraggio di una Set con i diversi predicati. (i predicati sono definiti nel file "set.h")

test_set_int_operatori() è il test che testa la concatenazione e l'intersezione tra due Set

test_set_string() è il test che esegue l metodi della classe Set costruita per il tipo String

test_set_point() è il test che esegue l metodi della classe Set costruita per il tipo Pointer (che è un oggetto creato). P.S. quando cerco di aggiungere un point nella Set questo mi da un errore in quanto, all'interno dell'add, c'è il controllo di uguaglianza per vedere se l'elemento è già inserito, che in questo caso fallisce perché il compilatore non sa in base a che criterio decidere se un punto è uguale ad un altro.

Per mancanza di tempo non sono riuscito ad ovviare al problema.