Relazione progetto C++
Febbraio 2022

Luca Poli [852027]

l.poli6@campus.unimib.it

Introduzione

Un set è una struttura dati dinamica dove non sono ammessi duplicati, accessibile in sola lettura e modificabile tramite apposite operazioni.

Per realizzare il dinamismo ho scelto di usare una lista doppiamente concatenata, raggiunta con un puntatore all'inizio e uno alla fine, questa struttura si basa sul nodo, ovvero l'elemento interno che la compone.

Perché proprio una lista e non un array dinamico?

Le principali operazioni su un set saranno la lettura totale, l'inserimento o la cancellazione di un elemento; una lista permette queste operazioni in tempistiche simili a quelle di un array, ma non presenta il problema di raggiungere la dimensione massima. In tal caso si dovrà provvedere a espanderlo copiandolo in un array più grande, perdendo però di efficienza.

L'unico svantaggio di usare una lista è nell'accesso casuale ad un nodo (fatto con l'operatore []).

Struttura dati interna: Il nodo

Possiamo dire che una lista è una serie di nodi concatenati fra loro, dove ogni nodo memorizza un singolo valore. Un nodo contiene tre dati: il valore, un puntatore al nodo successivo e un puntatore al nodo precedente. Ho scelto una doppia concatenazione per semplificare l'inserimento in coda e per permettere l'uso di un iteratore bidirezionale.

Il nodo presenta un'implementazione di base, caratterizzata dal costruttore base, il copy-constructor, l'operatore di assegnamento e una serie di costruttori secondari per ogni evenienza.

Struttura dati: Il set

La struttura in sé

Dunque il set è rappresentato da un puntatore di inizio e di fine alla coda di nodi, dalla una variabile che traccia la sua dimensione e da un funtore usato per determinare quando due elementi sono uguali

I metodi fondamentali di ogni classe

Come ogni classe abbiamo bisogno di alcuni metodi fondamentali come costruttori, distruttori, operatori e altri metodi per leggere i parametri interni.

Di costruttori abbiamo:

- il costruttore di default che inizializza la struttura vuota,
- il copy-constructor che inizializza il set copiando interamente ogni valore da l'altro set (non si limita a "linkarlo"),
- due costruttori secondari che inseriscono nel set degli elementi presi rispettivamente da un array statico o da una coppia di iteratori, scartando gli elementi doppi.

Il distruttore si limita ad invocare un metodo apposito che percorre la lista a ritroso de allocando ogni nodo.

Riguardo agli operatori abbiamo:

- l'assegnamento che crea una copia (per valore come il copy-constructor) del set passato e assegna la copia a sé stesso
- il confronto (==) che scorre la lista del set e per ogni elemento cerca se esso si ripete nell'altro set
- l'operatore ad accesso casuale [] che scorre la lista del set fino a raggiungere l'indice corretto
- l'operatore di stampa, che scorre la lista e stampa ogni elemento

Infine, abbiamo altri metodi:

- Di ricerca, di cui uno privato che scorre la lista fino a trovare l'elemento (usando il funtore) e restituendone il puntatore al nodo; e uno pubblico che si limita a chiamare il precedente e restituire un booleano
- Altri per ottenere la size del set

I metodi specifici del set

Essi sono rispettivamente la add (per aggiungere un elemento) e la remove (per rimuoverlo):

- Add: prima verifica la presenza del valore e in tal caso lancia l'eccezione apposita (spiegata meglio dopo) e termina; in caso contrario procede con l'inserimento in coda (la posizione è indifferente, tuttavia, si preferisce la coda per dare un maggiore senso di ordine cronologico)
    - La verifica viene fatta con il metodo di ricerca privato
    - L'inserimento in coda viene realizzato come in una qualunque lista doppia con puntatore finale
- Remove: prima effettua la ricerca del valore e in caso di assenza lancia l'eccezione apposita (spiegata meglio dopo) e termina; altrimenti effettua la sua cancellazione
    - La verifica viene fatta con il metodo di ricerca privato
    - La cancellazione procede come una normale cancellazione di nodo in una lista doppia

I metodi generici

Essi sono:

- Il filter_out: che usa una coppia di iteratori per individuare gli elementi che soddisfano la proprietà templata in un funtore aggiuntivo
    - Del funtore è necessario venga implementato l'operatore () che specifica la proprietà
- Concatenazione (+): copia il primo set nel set di output; e usa una coppia di iteratori sul secondo set per inserire ogni elemento nel set copiato
- Intersezione (-): usa una coppia di iteratori sul primo set per inserire ogni elemento, che è presente nel secondo, nel set di output

I tre metodi generici, per scelta, restituiscono un puntatore ad un nuovo set; cedendo la responsabilità di de allocazione al chiamante, questo perché la restituzione con copia sarebbe stata troppo onerosa in termini di efficienza.

Iteratore

Per garantire un maggior grado di libertà d'uso ho scelto di implementare un iteratore bidirezionale; esso è composto da due puntatori ai nodi (pnode, backnode), il primo punta al nodo corrente, il secondo punta al nodo letto all'iterazione precedente.

Queste due variabili denotano 3 stati logici dell'iteratore:

- Normal state: esso si può spostare sia in avanti che indietro; quindi, le operazioni di incremento e decremento sono realizzate spostando il backnode a pnode, e spostando pnode sul successivo (o precedente nel caso di decremento)
- End state: l'iteratore si trova in una situazione di end, quando pnode ha superato una delle due estremità, è quindi fuori dalla lista; tuttavia, è ancora possibile rientrare tramite l'operazione inversa (incremento se in testa-1, decremento se in coda+1), tramite l'ausilio di backnode.
- Dead state: l'iteratore si trova in stato di errore; si raggiunge questa situazione quando si usa l'iteratore in End state con l'operazione non corretta (decremento se in testa-1, incremento se in coda+1).
    - Nota: ho scelto a livello progettuale di permettere l'operazione non corretta, la responsabilità rimane dunque a chi lo usa.
        - (così come è possibile incrementare un iteratore di fine unidirezionale deve essere possibile farlo con quello bidirezionale)

Due iteratori si definiscono uguali quando puntano allo stesso nodo (i due pnode corrispondono)

La coppia di iteratori (begin, end) corrisponde ad un iteratore posto in testa alla lista, e un iteratore posto in end state

Gestione delle eccezioni

Per quanto riguarda le eccezioni ne ho usate di due tipi; la prima fa parate della libreria standard è l'eccezione out_of_range e viene lanciata quando si usa un indice errato nell'operatore [].

Invece il secondo tipo è stato definito da me, si chiama set_element_error ed è derivato dall'eccezione standard domain_error.

Presenta due costruttori, il primo che ne permette l'utilizzo generico come se fosse un'eccezione standard; il secondo ne permette l'utilizzo specifico al caso del set, tramite il coinvolgimento di un booleano. Esso permette la distinzione tra un errore di presenza (usato nella add) e uno di assenza (usato nella remove) e restituisce il relativo messaggio di errore standard.

Main_test

In questo file vengono testate le principali caratteristiche del set su 3 tipi di dati: interi, stringhe, e voci (di una rubrica); infatti nella prima parte del file sono definiti i funtori e alcuni dati di default che verranno usati nei test.

Nella seconda parte vengono definiti 3 metodi per separare le tre categorie di test:

- Test fondamentali: in cui si verificano il copy constructor, le operazioni di add, remove, find, gli operatori e le stampe tramite il metodo e l'iteratore
- Test generiche: in cui si testano i 3 metodi generici (filter_out, operatore+, operatore-)
- Test iteratori: in cui si testa il comportamento specifico degli iteratori in end-state
    - Nota: parte del codice è commentato perché genera i crash attesi e ricercati con tali istruzioni

Infine nella terza parte, ovvero nel main stesso, si creano i set e si richiamano i tre metodi definiti precedentemente.