Relazione Progetto C++
Motta Giacomo

11/02/2022

851887

g.motta22@campus.unimib.it

In generale

Ho deciso di svolgere il progetto implementando il set tramite una lista, visto che il set è una lista di elementi tutti dello stesso tipo; il tipo del set deve poter essere deciso dall'utente, quindi ho templato il set su un tipo generico T

Set

Ho implementato il set tramite una lista, da me definita, composta dalle seguenti **variabili**:

- *_head puntatore al primo elemento del set
- _size numero di elementi attualmente presenti nel set

e dai seguenti **metodi**:

- costruttore di default
- copy constructor
- operatore di assegnamento
- distruttore
- clear usata per svuotare il set
- add usata per aggiungere un elemento nel set
- remove usata per rimuovere un elemento dal set
- size ritorna il numero di elementi attualmente presenti nel set
- operator[ ] ritorna l'i-esimo valore presente nel set, con accesso in sola lettura
- operator== ritorna true se this e other sono uguali, altrimenti false

- costruttore tramite due iteratori in input che possono essere di qualsiasi tipo, quindi anche diverso da quello di this, questo cast viene delegato al compilatore
- operator<< operatore di stream per semplificare la stampa di un set
- begin ritorna l'iteratore all'inizio del set
- end ritorna l'iteratore alla fine del set

Element

I singoli elementi del set sono realizzati tramite una struct element, di supporto alla classe set, definita internamente alla classe set, questo in quanto non vi è motivo che venga esposto all'esterno il metodo con la quale è stato implementato il set.
La struct element è definita dalle seguenti variabili:

- value valore memorizzato di tipo T
- *next puntatore all'elemento successivo, se esiste

la struct è dotata dei quattro metodi fondamentali ed oltre ad essi ha tre costruttori secondari:

- `element(const T &v, element* n)` che permette di inizializzare con il valore che si preferisce l'elemento
- `explicit element(const T &v)` che permette di inizializzare solo il valore, lasciando il puntatore a next a nullptr
- `explicit element(element *n)` che permette di inizializzare il valore di next senza modificare il valore di value

Iteratori

Ho implementato l'iteratore tramite un forward iterator, in quanto ho deciso di implementare il set salvando solo il puntatore al next di un determinato elemento, questo perché leggendo il testo ed i vari metodi della classe set, non ho ritenuto necessario implementare una lista doppio linkata.
L'iteratore, come le altre classi, è dotato di tutti e quattro i metodi fondamentali, ed oltre ad essi ha anche:

- operator* ritorna il dato riferito dall iteratore
- operator-> ritorna il puntatore al dato riferito dall iteratore
- operator++ operatore di post-incremento
- operator++ operatore di pre-incremento
- operato== ritorna true se this e other sono uguali

oltre a ciò, il set viene posto friend della classe iteratore per fare in modo che possa utilizzare il costruttore privato dell'iteratore.

Funzioni globali

- operator+ : funzione globale che implementa la concatenazione tra due set, quindi ne ritorna il set risultante
- operator- : funzione globale che implementa l'intersezione tra due set, quindi ne ritorna il set risultante
- filter_out : funzione globale che seleziona tutti e soli quegli elementi del set che soddisfano il predicato passato in input, quindi ritorna il set risultante.

Main

Nel main ho testato tutte le funzioni della classe set, sia su tipi "classici": int e char, che su un tipo custom: point

Valgrind

l'esecuzione del main non riporta alcun errore di memoria, nello specifico il risultato di valgrind:

```
==95==
==95== HEAP SUMMARY:
==95==     in use at exit: 0 bytes in 0 blocks
==95==   total heap usage: 221 allocs, 221 frees, 77,232 bytes allocated
==95==
==95== All heap blocks were freed -- no leaks are possible
==95==
==95== For lists of detected and suppressed errors, rerun with: -s
==95== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```