Relazione progetto C++

Nome: Gabriele

Cognome: Ferrario

Matricola: 817518

Mail: g.ferrario@campus.unimib.it

- **Premessa:**

Ho scelto di utilizzare una lista semplice per rappresentare un set in quanto essendo una struttura dati dinamica mi consente di avere una gestione ottimale della memoria allocando man mano gli elementi (non ho un numero prefissato di elementi) e una gestione semplice dei dati.

- **Dati:**

Per rappresentare un Set generico ho scelto di appoggiarmi su una struct che definisce un Nodo.

Ogni elemento di un set è rappresentato da un Nodo che è composto da:

1- value: di tipo T che rappresenta il dato (il tipo è generico, quindi verrà associato il tipo passato durante la dichiarazione del set).
2- next: di tipo Nodo ed è un puntatore all'elemento successivo del Set.

Il campo value è dichiarato const nei metodi in cui è passato come parametro in ingresso, così una volta scelto il valore non potrà essere modificato.

Un set generico è rappresentato tramite una lista semplice, quindi la classe set prevede un attributo di tipo Nodo chiamato _head che è il puntatore al primo elemento della lista e inoltre è dotata di un oggetto _equal. Quest'ultimo è il funtore che la classe usa per svolgere i confronti. In quanto un set non deve contenere elementi uguali.

Nel set.h ho definito un funtore di confronto generico per tipi standard che valuta semplicemente l'uguaglianza tra due elementi ritornando il risultato del confronto == .

Nella classe set.h sono state definite due classi per gestire due tipi di eccezione.

- **Implementazione:**

Ho implementato due costruttori per la classe Nodo:

1- Costruttore di default che istanzia un nodo vuoto, settando il campo next a null.
2- Costruttore secondario che inizializza un nodo tramite una variabile valore (v) e un eventuale puntatore all'elemento successivo (default settato a 0).

Metodi fondamentali per la classe Set:

1- Costruttore di default che istanzia un set vuoto, settando _head a null.
2- Costruttore di copia che istanzia un set con i valori presenti nel set che il metodo riceve in ingresso. Gli elementi sono copiati utilizzando una lista temporanea tmp che punta alla testa del set passato al metodo come input. La lista temporanea è fatta scorrere da un while e ad ogni ciclo viene aggiungo l'elemento corrente al puntatore this (puntatore all'istanza che ha invocato il metodo). Questo metodo potrebbe sollevare un'eccezione dovuta all'allocazione della memoria.
3- Ridefinizione dell'operatore di assegnamento =, il quale consente la copia tra set. Si appoggia sul costruttore di copia e sulla swap. Potrebbe sollevare un'eccezione dovuta all'allocazione della memoria.
4- Distruttore che serve a liberare la memoria allocata. Questo metodo si appoggia su un metodo chiamato clear;

Altri metodi presenti nella classe Set:

1- Clear: è il metodo che si occupa di cancellare il contenuto del set, liberando anche la memoria. Ogni nodo del set è cancellato tramite una delete che viene applicata su ogni elemento del set utilizzando un while per scorrere la struttura.
2- Add: permette di inserire un nuovo elemento nel set verificando che l'elemento che si sta cercando di inserire non sia presente. Inizialmente viene creato un nodo contenente il valora da inserire nel set. L'inserimento avviene verificando se la lista è vuota (head==NULL) e in questo caso viene inserito in testa il nodo. Altrimenti viene eseguito un while con lo scopo di scorrere la lista verificando se non esiste un valore uguale utilizzando il funtore _equal. Se il valore non è presente viene inserito in coda. Se l'elemento è già contenuto nel set viene sollevata un'eccezione equal_element_exception. Questa eccezione è implementata tramite una classe. Inoltre in questo caso eseguo una delete sul nodo non inserito per liberare la memoria.
3- Conta_elementi: è un metodo di supporto usato per contare il numero di elementi presenti nel set. Consiste in un while che usa un contatore per contare il numero di elementi e ritorna il contatore.

4- Ridefinizione del operatore []. Questo metodo usa conta_elementi per verificare se si cerca di accedere ad una posizione corretta quindi con indice minore del numero di elementi. Per verificare questo fatto ho usato un'asserzione. Questo metodo consente l'accesso in sola lettura e tramite un ciclo for scorre fino alla posizione cercata ritornando il valore desiderato. (gli indici iniziano da 0, quindi il primo elemento è alla posizione 0)
5- Remove: permette di rimuovere un valore dal set. Inizialmente verifico se il set non è vuoto. Successivamente analizzo la testa e vedo se contiene il valore da eliminare, in caso affermativo elimino il nodo e sostituisco la testa. Altrimenti uso un ciclo while per cerca l'elemento, se lo trovo elimino il nodo e lo sostituisco con il successivo. In caso contrario (se non trovo il valore) sollevo un'eccezione not_found_element_Exception, questo è possibile tramite una variabile booleana che è settata a false ma assume valore true nel caso in cui l'elemento venga trovato. L'eccezione sollevata è implementata tramite una classe.
6- Un costruttore secondario che consente di creare un set tramite l'uso di una coppia di iteratori che fanno riferimento ad una sequenza di dati. Potrebbe sollevare un'eccezione.
7- Sono stati implementati gli iteratori costanti definendo i Traits. Per gli iteratori sono stati implementati i metodi fondamentali, quindi costruttore di default, costruttore di copia e ridefinizione dell'operatore di assegnamento. Inoltre sono stati implementati gli operatori per interagire con l'iteratore come l'operatore di dereferenziamento. Infine ci sono i metodi begin e end. Il primo ritorna l'iteratore all'inizio della sequenza mentre il secondo l'iteratore alla fine della sequenza.
8- Ridefinizione del operatore di stream << che consente la stampa dei set. Il metodo usa due iteratori utilizzati all'interno di un for per scorrere la sequenza dei dati creando l'oggetto di stream di cui ne verrà ritornato il reference.

-Main:

Il main inizialmente è composto da due funzioni globali:

1- Filter_out: è una funzione globale e generica che riceve in ingresso un set generico S e un predicato booleano P, ritornando un set formato dagli elementi del set S che non soddisfano il predicato P. La funzione è stata implementata definendo due iteratori uno che fa riferimento all'inizio della sequenza e uno alla fine e un set generico sf. Questi iteratori

vengono usati in un for che scorre la sequenza di dati aggiungendo a sf gli elementi che non rispettano il predicato tramite un add, in fine sf viene ritornato tramite return.

2- Ridefinizione dell'operatore + che consente concatenazione tra due set. Riceve in ingresso due set e ne restituisce uno formato dai loro elementi. L'ho implementato definendo un set generico s a cui ho assegnato i valori del primo set passato in input tramite operatore di assegnamento. Infine con gli iteratori sono andato ad aggiungere uno alla volta i valori del secondo set utilizzando l'add. La funzione può sollevare un'eccezione dovuta alla presenza di almeno un elemento uguale tra i due set da concatenare.

Il main inoltre contiene cinque test, i primi tre sono applicati a dei set di interi, il quarto test utilizza set di stringhe, mentre l'ultimo usa set definiti su una struttura che definisce un punto in due dimensioni.

Per ogni tipo di dato ho definito un funtore di confronto booleano, dei funtori per filtrare i dati, mentre per i punti oltre a queste cose ho definito l'operatore di stream <<.

I test sono composti principalmente da stampe a schermo tranne per alcuni test che ho usato le asserzioni.

Nei test ho cercato di catturare tutte le criticità che si potevano verificare e in caso di sollevamento di eccezione ho pensato di catturarle e di gestirle fornendo una stampa di errore per segnalare l'evento.