Corso di Algoritmi e Strutture Dati
Esercitazione 5: alberi binari di ricerca

Per rendere più efficiente la propria gestione amministrativa, una azienda ha deciso di prendere in considerazione un semplice sistema che mantiene un dizionario (elenco ordinato) dei dipendenti. Per semplicità l'azienda ha deciso di riciclare la struct `Impiegato` definita nella scorsa esercitazione 1.

Nella esercitazione 1 l'elenco era realizzato mediante una lista ordinata. Tuttavia, la ricerca di un dipendente in tale lista ha sempre complessità $O(N)$, non accettabile quando il numero di dipendenti diventa grande. Perciò l'azienda ha optato per una struttura dati che può essere più efficiente: un albero binario di ricerca. Naturalmente con l'albero binario di ricerca la complessità del caso pessimo resta $O(N)$, ma l'azienda confida che il caso pessimo non si presenti, e che l'albero sia e rimanga accettabilmente bilanciato.

Si realizzi, incapsulandolo in una struct C++, un albero binario di ricerca con elementi di tipo `Impiegato`. La relazione di ordine tra elementi di tipo `Impiegato` è quella definita nella scorsa esercitazione 1. L'albero binario di ricerca, ovviamente, fa uso di quella relazione di ordine per inserire elementi rispettando il proprio invariante (ossia: ciascun elemento nel sottoalbero sinistro è minore o uguale rispetto alla radice, e ciascun elemento nel sottoalbero destro è maggiore rispetto alla radice).

La rappresentazione dell'albero binario dovrà essere di tipo collegato, e ogni nodo deve avere due puntatori ai figli (figlio sinistro e figlio destro). Non sono ammesse realizzazioni indicizzate (es. con array). È permesso utilizzare la libreria standard del C++, ad eccezione della STL (Standard Template Library). Non è permesso ricorrere ad altre librerie.
Le operazioni dell'albero binario di ricerca da realizzare come metodi della struct sono le seguenti:

- inserimento di un elemento nell'albero; l'elemento da inserire è uno dei parametri del metodo
- visita ordinata (che poi è la visita in profondità simmetrica) finalizzata a visualizzare l'intero elenco ordinato
- ricerca di un elemento nell'albero; poichè in questo scenario molto semplificato l'elemento e la sua chiave coincidono (non esistono informazioni aggiuntive, oltre al cognome, al nome e all'anno di assunzione), basterà indicare come parametro del metodo l'elemento da cercare (di tipo Impiegato) e restituire come risultato un semplice booleano (true se e solo se l'elemento è stato trovato)
- cancellazione di un elemento dall'albero, con conseguente riorganizzazione di quest'ultimo; l'elemento da cancellare è uno dei parametri del metodo e il risultato è di tipo booleano (true se e solo se l'elemento è stato cancellato; si tenga presente che un elemento non può essere cancellato se non era già presente nell'albero).

La struct C++ che realizza l'albero e le sue operazioni, unitamente alla struct `Impiegato` su cui essa si basa, deve essere impiegata in un semplice programma C++. Tale programma deve svolgere, nell'ordine, le azioni seguenti:

1. acquisire dall'esterno un elenco di impiegati, seguendo le specifiche illustrate nella scorsa esercitazione 1, e man mano inserirli nell'albero

2. un ciclo che permetta la scelta ripetuta di una tra le azioni seguenti:
    - visualizzare l'elenco ordinato degli impiegati inseriti
    - ricercare un impiegato, acquisendo dall'esterno i suoi dati identificativi e visualizzando il risultato dell'operazione
    - cancellare un impiegato, acquisendo dall'esterno i suoi dati identificativi e visualizzando il risultato dell'operazione
    - uscire dal programma.