ESERCIZIO 2
- ALBERI BINARI DI DATI GENERICI -

F. MOGAVERO

Implementare due librerie di funzioni in **Linguaggio C++** per la gestione di strutture dati dinamiche di tipo **albero binario** e **albero binario di ricerca** contenenti **dati generici**, ovvero interi, stringhe, ecc. In particolare, la prima struttura dovrà essere implementata sfruttando le seguenti due **rappresentazioni concrete**: (1) **vettore dei nodi**; (2) **puntatori a nodi**. La seconda sfrutterà, invece, la sola rappresentazione esplicita di un albero binario per mezzo di **puntatori a nodi**.

Le funzionalità da realizzare sono di seguito elencate:

(1) **costruzione** e **distruzione** di una struttura dati;
(2) operazioni di **assegnamento** e **confronto** tra istanze diverse della specifica struttura dati (per quanto riguarda gli alberi binari arbitrari, il confronto deve poter essere effettuato indipendentemente dalla rappresentazione concreta);
(3) operazioni comuni alle due strutture dati: **controllo di esistenza** di un dato valore; operazioni di **attraversamento** e **accumulazione di un valore** (in ampiezza/pre-ordine/ordine/post-ordine) (funzioni **traverse** e **fold**); test di **vuotezza**; lettura della **dimensione**; **svuotamento** della struttura; **interrogazione** delle proprietà di un nodo (**accesso in lettura/scrittura** al dato, **controllo di esistenza e accesso** al figlio sinistro/destro); **navigazione** per mezzo di iteratori;
(4) funzioni specifiche degli alberi binari: **applicazione di una funzione (in ampiezza/pre-ordine/ordine/post-ordine)** a tutti gli elementi (**funzioni map**);
(5) funzioni specifiche degli alberi binari di ricerca: **inserimento/eliminazione** di un dato elemento; **rimozione, rimozione con lettura** e **lettura non distruttiva** dell'elemento minimo/massimo o del predecessore/successore di un dato elemento.

Al fine di poter testare adeguatamente il funzionamento delle librerie sopra descritte, si richiede di definire (esternamente alle stesse, in un opportuno file di test richiamato dal "main") un insieme di procedure di **test unitario**. In aggiunta a questo, è necessario prevedere l'accesso alla funzionalità di test prevista dal docente.

Il codice sorgente prodotto dovrà seguire pedissequamente (sia nei nomi delle funzioni di libreria, sia nella strutturazione, gerarchia di classi e nei nomi delle diverse directory e file ".cpp" e ".hpp") la forma riportata nel template Exercise2.zip associato al presente esercizio.

Per lo sviluppo delle librerie, in aggiunta alle spiegazioni del docente e al diagramma delle classi reperibile sul sito docenti d'ateneo, si faccia riferimento alle sezioni 5.1, 5.2, 5.3 e 5.4 di [1].

RIFERIMENTI BIBLIOGRAFICI

[1] Clifford A. Shaffer. *Data Structures and Algorithm Analysis*, Edition 3.2 (C++ Version), 2013.