ESERCIZIO 3
- ALBERI BINARI CON RAPPRESENTAZIONI MULTIPLE -

F. MOGAVERO

Implementare una libreria di funzioni in **Linguaggio C++** per la gestione di una struttura dati dinamica di tipo **albero binario** contenente **dati generici**, ovvero interi, float, stringhe, struct, ecc. Tale struttura dovrà essere implementata sfruttando le seguenti due **rappresentazioni concrete**, con efficiente **ridimensionamento automatico** dello spazio allocato: (1) **vettore dei nodi**; (2) **puntatori ai 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 struttura dati (il confronto deve poter essere effettuato indipendentemente dalla rappresentazione concreta);
(3) operazioni specifiche della struttura dati (**interrogazione** delle proprietà di un nodo: **accesso in lettura/scrittura** al dato contenuto nei nodi, **controllo di esistenza e accesso** al padre, figlio sinistro/destro e fratello sinistro/destro di un nodo; **inserimento** di un nuovo figlio sinistro/destro di un dato nodo; **rimozione** di un sottoalbero sinistro/destro di un dato nodo);
(4) operazioni generiche sulla struttura dati (**controllo di esistenza** di un dato valore; operazioni di **applicazione di una funzione (in ampiezza/pre-ordine/ordine/post-ordine)** a tutti gli elementi: **funzioni map**; operazioni di **accumulazione di un valore (in ampiezza/pre-ordine/ordine/post-ordine)** estratto dagli elementi: **funzioni fold**; test di **vuotezza**; lettura della **dimensione**; **svuotamento** della struttura).

Al fine di poter testare adeguatamente le librerie sopra descritte, si richiede di definire (esternamente alle stesse, in un opportuno file di test richiamato dal "main") un insieme di procedure che implementi le seguenti funzionalità:

(1) **scelta dell'implementazione** (vettore vs puntatori) e del **tipo di dati** da gestire (interi, ecc.);
(2) **popolamento della struttura** precedentemente scelta con n valori del tipo opportuno generati casualmente, dove n è un parametro dato dall'utente in ingresso (la forma dell'albero deve essere anch'essa generata in modo casuale);
(3) **visualizzazione in ampiezza/pre-ordine/ordine/post-ordine di tutti gli elementi** (effettuata per mezzo dell'opportuna **funzione map**);
(4) **controllo di esistenza** di un dato valore;
(5) **calcolo di una delle seguenti funzioni** (effettuato per mezzo delle **funzioni fold**) e relativa visualizzazione del risultato: prodotto per gli interi minori di n, somma per i float maggiori di n, e concatenazione per le stringhe con lunghezza minore o uguale a n, dove n è un parametro dato dall'utente in ingresso;
(6) **applicazione di una delle seguenti funzioni** a tutti gli elementi: $3n$ per gli interi, n^3 per i float, e concatenazione in testa di una specifica stringa *str* data dall'utente in ingresso nel caso delle stringhe.

Da un opportuno menu, dovrà essere inoltre possibile operare sulla struttura scelta attraverso le funzioni di libreria di cui al punto (3). Infine, 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 Exercise3.zip associato al presente esercizio.