

ESERCIZIO 4

- HEAP E PRIORITY QUEUE DI DATI GENERICI -

F. MOGAVERO

Implementare due librerie di funzioni in **Linguaggio C++** per la gestione di strutture dati dinamiche di tipo **heap** e **priority queue** contenenti **dati generici**, ovvero interi, float, stringhe, struct, ecc. Tali strutture dovranno essere implementate sfruttando la rappresentazione implicita di un albero binario per mezzo del **vettore dei 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;
- (3) operazioni specifiche della struttura dati (**ordinamento** dei dati; **inserimento** di un dato elemento; **rimozione**, **rimozione con lettura** e **lettura non distruttiva** dell'elemento immediatamente accessibile; **cambio di priorità** di un elemento);
- (4) operazioni generiche sulla struttura dati (**controllo di esistenza** di un dato valore; operazioni di **applicazione di una funzione (in pre-ordine/post-ordine)** a tutti gli elementi: **funzioni map**; operazioni di **accumulazione di un valore (in pre-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 della struttura** (heap vs priority queue) 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;
- (3) **visualizzazione in pre-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: $n * (-1)^n$ per gli interi, $-n$ 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 Exercise4.zip associato al presente esercizio.