Esame di Programmazione II

Appello di giorno 11 Febbraio 2015
Università degli Studi di Catania - Corso di Laurea in Informatica

Testo della Prova

Definizione Iniziale.
Una *table* è una struttura dati lineare in cui ogni elemento ha un successore ed un predecessore, ad esclusione del primo e dell'ultimo. Se T è una table e $\langle x_1, x_2, \ldots, x_n \rangle$ è la sequenza degli elementi memorizzati al suo interno, allora x_1 è il primo elemento di T, x_2 è il suo secondo elemento, mentre x_n è l'ultimo elemento di T. Diremo che la dimensione di T è n.

Specifiche.
La corretta implementazione di ciascuno dei seguenti esercizi permette l'acquisizione di 9 punti. La corretta implementazione della classe come template è facoltativa e permette l'acquisizione di ulteriori 3 punti:

1. Si fornisca una classe `C++`, denominata `MyTable<H>`, che implementi la seguente interfaccia `Table<H>`, che rappresenta una table e contenente i seguenti metodi virtuali. La struttura dati potrà essere implementata attraverso una lista concatenata o attraverso un array.

   - `void del(H` x`)` cancella dalla table l'elemento di valore x, se presente. Nel caso fossero presenti più elementi con lo stesso valore, la procedura dovrà cancellare l'ultimo elemento uguale a x.
   - `void insert(H` x`)` inserisce all'interno della table il nuovo elemento x, anche se questo è già presente nella struttura dati.
   - `H* max()` restituisce un puntatore all'elemento di valore più grande contenuto nella struttura dati. Restituisce `null` nel caso in cui la table sia vuota;
   - `H* min()` restituisce un puntatore all'elemento di valore più piccolo contenuto nella struttura dati. Restituisce `null` nel caso in cui la table sia vuota;
   - `void print()` stampa la lista degli elementi della struttura dati. Gli elementi della table dovranno essere racchiusi tra parentesi quadre.

   Si crei quindi un'istanza di `MyTable<int>` e si inseriscano al suo interno i valori `3, 7, 1` e `8` utilizzando la procedura INSERT. Si inseriscano in seguito i valori `5,2,6,1` e `8` utilizzando ancora la procedura `insert`. Infine si eliminino gli elementi `8,7` e `1`. Si invochino in seguito le procedure `max`, `min` e `print`. L'output sarà quindi il seguente:

   ```
   8
   1
   [3,1,8,5,2,6]
   ```

2. Si fornisca una classe `C++`, denominata `OrderedTable<H>`, che implementi una table in cui gli elementi sono memorizzati in modo ordinato. Se $\langle x_1, x_2, \ldots, x_n \rangle$ è la sequenza degli elementi memorizzati al suo interno, dovrà valere la relazione $x_1 \leq x_2 \leq \ldots \leq x_n$.

Si effettuino su tale struttura dati le stesse operazioni proposte nel punto precedente. In tal caso l'output sarà il seguente:

```
8
1
[1,2,3,5,6,8]
```

3. Si fornisca una classe `C++`, denominata `CircularTable<H>`, che implementi una table in cui gli elementi sono memorizzati in modo circolare non ordinato. In tale struttura dati ogni elemento memorizzato al suo interno avrà quindi un successore ed un predecessore. La table dovrà anche contenere un puntatore all'elemento più *vecchio* presente nella struttura dati, cioè l'elemento che è stato inserito prima di tutti gli altri elementi presenti nella struttura.
   Si crei quindi un'istanza di `CircularTable<int>` e si inseriscano al suo interno i valori `3, 7, 1` e `8` utilizzando la procedura di inserimento. Si inseriscano in seguito i valori `5, 2, 6, 1` e `8` utilizzando ancora la stessa procedura. Infine si eliminino gli elementi `8, 7` e `1`. Si invochi in seguito la procedura `print`. L'output stamperà i seguenti elementi (non necessariamente nell'ordine dato):

   ```
   [3,1,8,5,2,6]
   ```

```
template <class H> class Table {
  public:
     virtual void del(H x) = 0;
     virtual void insert(H x) = 0;
     virtual H* max() = 0;
     virtual H* min() = 0;
     virtual void print() = 0;
 }
```