Operatii extra laborator SDA 30.04.2025

TAD Matrice
// creaza un iterator asupra tuturor elementelor din Matrice (indiferent dacă acestea sunt sau nu NULL_TELEM). Iteratorul va returna elementele în ordine parcurgerii pe linii (mai întâi elementele primei linii, apoi ale celei de-a doua linie, etc).
IteratorMatrice iterator() const;

Obs: Trebuie implementată clasa IteratorMatrice, de asemenea.

TAD Colecție
// elimină toate apatițiile elementului *elem* din colecție
// returnează numărul de elemente eliminate
int eliminăToateAparițiile(TElem elem);

TAD Mulțime
// adaugă toate elementele din mulțimea b în mulțimea curentă
void reuniune(const Multime& b);

TAD Listă
// elimină din lista curentă toate elementele care apar în *lista* (lista argument)
// returnează numărul de elemente eliminate
int eliminaToate(Listă& lista);

TAD ListăOrdonată
// elimină din lista curentă toate elementele care apar în *lista* (lista argument)
// returnează numărul de elemente eliminate
int eliminaToate(ListăOrdonată**& lista);**

TAD Dicționar
Transformați iteratorul pentru a putea elimina elementul curent. Adăugați operația următoare în clasa IteratorDicționar:
// elimină și returnează perechea curentă referită de iterator
// după operație, perechea curentă de iterator este elementul următor dicționarului, sau, în cazul în care elementul eliminat a fost ultimul, iteratorul este nevalid
// aruncă excepție în cazul în care iteratorul este nevalid
TElem elimina ();
Obs: Pentru ca această operație să funcționeze, trebuie să efectuați câteva alte modificări în cod:

- operația iterator din interfața dicționarului nu mai este const
- Referința la dicționar a iteratorului nu mai este const (dar este încă o referință)
- Parametrul transmis constructorului clasei care implementează iteratorul nu mai este const

TAD DicționarOrdonat

Transformați iteratorul pentru a putea elimina elementul curent. Adăugați operația următoare în clasa IteratorDicționar:

// elimină și returnează perechea curentă referită de iterator

// după operație, perechea curentă de iterator este elementul următor dicționarului, sau, în cazul în care elementul eliminat a fost ultimul, iteratorul este nevalid

// aruncă excepție în cazul în care iteratorul este nevalid

TElem elimina ();

Obs: Pentru ca această operație să funcționeze, trebuie să efectuați câteva alte modificări în cod:

- operația iterator din interfața dicționarului nu mai este const
- Referința la dicționar a iteratorului nu mai este const (dar este încă o referință)
- Parametrul transmis constructorului clasei care implementează iteratorul nu mai este const

TAD Multidicționar

Transformați iteratorul pentru a putea elimina elementul curent. Adăugați operația următoare în clasa IteratorMultidictionar:

// elimină și returnează perechea curentă referită de iterator

// după operație, perechea curentă de iterator este perechea următoare din Multidicționar, sau, în cazul în care perechea eliminată a fost ultima, iteratorul devine nevalid

// aruncă excepție în cazul în care iteratorul este nevalid

TElem elimina ();

Obs: Pentru ca această operație să funcționeze, trebuie să efectuați câteva alte modificări ale codului:

- operația iterator din multidicționar nu mai este *const*
- Referința la multidicționar din iterator nu mai este *const* (dar este încă o referință)
- Parametrul transmis constructorului clasei care implementează iteratorul nu mai este *const*

TAD MultidicționarOrdonat

Transformați iteratorul pentru a putea elimina elementul curent. Adăugați operația următoare în clasa IteratorMultidictionar:

// elimină și returnează perechea curentă referită de iterator

// după operație, perechea curentă de iterator este perechea următoare din Multidicționar, sau, în cazul în care perechea eliminată a fost ultima, iteratorul devine nevalid

// aruncă excepție în cazul în care iteratorul este nevalid

TElem elimina ();

Obs: Pentru ca această operație să funcționeze, trebuie să efectuați câteva alte modificări ale codului:

- operația iterator din multidicționar nu mai este *const*
- Referința la multidicționar din iterator nu mai este *const* (dar este încă o referință)
- Parametrul transmis constructorului clasei care implementează iteratorul nu mai este *const*

TAD CoadăCuPriorități

// returnează prioritatea elementului *elem*. În cazul în care *elem* apare de mai multe ori, se va returna prioritatea uneia dintre apariții

//daca *elem* nu este în coadă, se returnează -1.

TPrioritate prioritateaElementului(TElem elem) const;

TAD VectorDinamic

Transformați iteratorului pentru a putea elimina elementul curent. Adăugați operația următoare în clasa Iterator:

// elimină și returnează elementul curent referit de iterator

// după efectuarea operației, elementul curent referit de iterator este următorul element din Vectorul Dinamic, sau, în cazul în care elementul eliminat a fost ultimul, iteratorul devine nevalid

// aruncă excepție în cazul în care iteratorul este nevalid

TElem elimina ();

Obs: Pentru ca această operație să funcționeze, trebuie să efectuați alte câteva modificări ale codului:

- Operația *iterator* din VectorDinamic nu mai este *const*
- referința la VectorDinamic în iterator nu mai este *const* (dar este încă o referință)
- Parametrul transmis la constructorul clasei Iterator nu mai este *const*

TAD Coadă

// returneaza elementul maxim din coadă (presupunem că elementele sunt numere întregi)

// aruncă excepție în cazul în care coada este goală

TElem maxim() const;