

Operatii extra laborator SDA 14.05.2025

TAD Matrice

// creaza un iterator asupra tuturor elementelor matricei de pe o anumita linie (indiferent dacă acestea sunt sau nu NULL_TELEM)

// aruncă excepție în cazul în care linia nu este o linie validă

IteratorMatrice iterator(linia int) const;

Obs: Va trebui implementată clasa IteratorMatrice, de asemenea.

TAD Colecție

//returnează numărul elementelor unice din colecție

int numărăElementeUnice() const;

TAD Mulțime

// păstrează în mulțime numai acele elemente care respectă condiția dată

void filtreaza(Condiția cond);

Condiția este o funcție care primește ca parametru un element și returnează adevărat sau fals (adevărat dacă elementul respectă condiția și fals, altfel)

Obs: adăugați următorul typedef la începutul fișierului Multime.h

typedef bool (***Condiție**)(TElem);

TAD Listă

Transformați iteratorul pentru a putea face k pași în urmă. Adăugați operația următoare în clasa IteratorListă:

// mută cursorul iteratorului a.î. să refere al k-lea element în urmă, începând de la cel curent. Iteratorul devine nevalid în cazul în care există mai puțin de k elemente anterioare elementului curent în listă.

// aruncă excepție în cazul în care iteratorul este nevalid sau k este zero ori negativ.

void revinoKPasi(int k);

TAD ListăOrdonată

Transformați iteratorul pentru a putea face k pași în urmă. Adăugați operația următoare în clasa IteratorListă:

// mută cursorul iteratorului a.î. să refere al k-lea element în urmă, începând de la cel curent. Iteratorul devine nevalid în cazul în care există mai puțin de k elemente anterioare elementului curent în listă.

// aruncă excepție în cazul în care iteratorul este nevalid sau k este zero ori negativ.

void revinoKPasi(int k);

TAD Dicționar

// returneaza valoarea care apare cel mai frecvent în dicționar. Dacă mai multe valori apar cel mai frecvent, se returnează una (oricare) dintre ele.

// Dacă dicționarul este vid, operația returnează NULL_TVALOARE

TValoare ceaMaiFrecventaValoare() const;

TAD DicționarOrdonat

// returneaza valoarea care apare cel mai frecvent în dicționar. Dacă mai multe valori apar cel mai frecvent, se returnează una (oricare) dintre ele.

// Dacă dicționarul este vid, operația returnează NULL_TVALOARE

TValoare ceaMaiFrecventaValoare() const;

TAD Multidicționar

// returneaza valoarea care apare cel mai frecvent în multidicționar. Dacă mai multe valori apar cel mai frecvent, se returnează una (oricare) dintre ele.

// Dacă multidicționarul este vid, operația returnează NULL_TVALOARE

TValoare ceaMaiFrecventaValoare() const;

TAD MultidicționarOrdonat

// returneaza valoarea care apare cel mai frecvent în multidicționar. Dacă mai multe valori apar cel mai frecvent, se returnează una (oricare) dintre ele.

// Dacă multidicționarul este vid, operația returnează NULL_TVALOARE

TValoare ceaMaiFrecventaValoare() const;

TAD CoadăCuPriorități

Această operație nu ar trebui să fie adăugată în clasa CoadăCuPriorități. Ar trebui să fie implementată în fișierul corespunzător programului principal folosind operațiile din interfața de CoadăCuPriorități, fără a modifica implementarea clasei și fără accesarea reprezentării.

// tipărește conținutul unui container CoadăCuPriorități. La sfârșitul operației, coada tipărită conține toate elementele pe care le conținea înainte, în aceeași ordine.

void tipăreșteCoadăCuPriorități(CoadăCuPriorități& cp);

TAD VectorDinamic

// elimină toate elementele între două poziții i și j

// aruncă excepție în cazul în care i sau/și j nevalid(e)

void eliminaIntre(int i, int j);

TAD Coadă

Această operație nu ar trebui să fie adăugată în clasa Coadă. Ar trebui să fie implementată în fișierul corespunzător programului principal folosind operațiile din interfața de Coadă, fără a modifica implementarea clasei și fără accesarea reprezentării.

// tipărește conținutul unui container Coadă. La sfârșitul operației, coada tipărită conține toate elementele pe care le conținea înainte, în aceeași ordine.

void tipăreșteCoadă (Coadă& cp);