GESZTI BENCE
WXNDSZ

Feladat

Ábrázoljon egy négyzetes ritka mátrixot láncolt ábrázolással úgy, hogy a mátrix minden nem nulla elemét tartalmazó listaelem két láncolt listába is be legyen fűzve: a mátrix adott sorát és adott oszlopát reprezentáló listába! A sorokat reprezentáló listák az oszlopindexek szerint, az oszlopokat reprezentáló listák sorindex szerint rendezve tárolják a mátrix elemeket! Valósítsa meg a mátrix adott indexű elemének kiolvasását és felülírását elvégző műveletet a zárójel operátorral! Készítsen olyan műveletet, amely legfeljebb n*(n-1)/2 lépésben eldönti, hogy a mátrix diagonális-e! Írja meg a mátrixot kiíró operátor<<-t!

Matrix: n*n -es ezt a size valtozo fogja tarolni. Lesz két tömb _row és _column a listak fejelemeit fogja tartalmazza.
Elem modosítasa: matrix(sor, oszlop, érték) két fv vel végezzük el: addElement(sor, oszlop, érték) : ami beszur egy adott helyre egy értéket és aremoveElement(sor, oszlop) torli az adott helyen lévő elemet ha van elem.
Elem kiírasa: matrix(sor, oszlop) vissza adja az adott sor és oszlopon lévő elem értékét ezt fogjuk kiírni.

Matrix kiírasa: << operator fogja végezni, amely felsorolja a matrixunknak minden értékét.

Diagonalis-e?: Matrixnal azt jelenti hogy csak a főátloban létezik érték a többi elem mind nulla ekkor diagonalis a matrix. Tehát meg nézzük az i. dik sor ban hogy van-e elem és ha van akkor megnézzük hogy annak a place komponense i - e , ha igen és nincs tobb elem a sorban akkor diagonalis. ha meg találjuk az elso olyan elemet aminek a place komponense nem egyenlő az i vel le is állunk és hamis értékkel térünk vissza. Ha az átlo mindegyik pontján van elem akkor maximum 2n a müvelet igény, ami sokkal kevesebb mint az n*(n-1)/2.

```
bool Matrix::isDiagon()
{
   for(int i=0;i<size;i++)
   {
      Node* p =&_row[i];
      p=p->next;
      while (p!=NULL) {
         if (p->place != i+1) return false;
         p=p->next;
      }
   }
   return true;
}
```

Matrix.hpp

```
#ifndef matrix_hpp
#define matrix_hpp
#include <iostream>

struct Node{
    int value,place;  //Soroknak az oszlop index, Oszlopoknál a sor index
    Node *next;

    Node(int k=0 ,int a=0, Node* p=NULL):
    place(k),value(a), next(p) {}
};

class Matrix
{
public:
    enum Exeptions{OVERINDEX};

    Matrix(int n);
    ~Matrix();

    bool isDiagon();

    void operator() (const int &row, const int &column, const int &a) throw
(Exeptions);
    const int operator() (const int &row, const int &column) throw (Exeptions);
    friend std::ostream& operator << (std::ostream& s, const Matrix& matrix);

private:

    int size;
    Node *_row, *_column;

    void addElement(const int &row, const int &column, const int &a);
    void removeElement(const int &row, const int &column);


};


#endif /* matrix_hpp */
```

Konstruktor

Matrix::Matrix(int n)

Létre hozza az n*n es mátrixunkat.

Destruktor

Matrix::~Matrix()

Törli a sor és oszlop lista minden elemet majd azok tombjeit.

Elem modositas

void Matrix::addElement(const int &row, const int &column, const int &a)

Megváltoztatja a megadott sor és oszlopban lévő értékét.

Elem torlese

void Matrix::removeElement(const int &row, const int &column)

Torol egy elemet a sor és oszlop listákból, ha nem volt azon a helyen elem nem történik semmi.

Zárojel-1 operator

void Matrix::operator() (const int &row, const int &column, const int &a) throw (Exeptions)

Elem modositasat fogja megkönnyíteni. *Figyeli, hogy ne adjon meg nagyobb indexet.*

Zárojel-2 operator

const int Matrix::operator()(const int&row, const int &column) throw (Exeptions)

Egy adott elem értéket fogja vissza adni. *Figyeli, hogy ne adjon meg nagyobb indexet.*

Diagonális-e?

bool Matrix::isDiagon()

Eldonti, hogy a matrixunk diagonális-e.

Kiíro operator

std::ostream& operator << (std::ostream& s, const Matrix &matrix)

Kiírja a képernyőre a matrixunkat.

Tesztelési terv

Hozzunk létre egy 10*10 es mátrixot
Valtoztassuk meg az értékét!

- **Adjunk meg érvényes helyet nézzük meg, hogy változott-e a matrix.**
- **Adjunk meg olyan helyet ami nincs a matrixban sor>10 v oszlop>10**
- **//Elvaras hibat dob.**

Elem kiírasa

- **Valasszunk egy érvényes sor oszlop indexet // elvaras megjelenik annak értéke**
- **Valasszunk egy mátrixon kivüli indexeket // elvaras hibat dob**

Diagonalis-e?

- **Nézzük meg nullmátrixra //elvaras igaz**
- **Töltsuk fel az átló elemeit //elvaras igaz**
- **Adjunk az atlon kivulre is elemet // elvaras hamis**