Graf dirigit amb multillistes d'adjacència. Insereix arestes.X95633_ca

Donada la classe *graf* que permet gestionar grafs dirigits i no etiquetats amb n vèrtexs (els vèrtexs són enters dins l'interval $[0, n-1]$), cal implementar el mètode

```
void insereix (nat orig, nat dest);
// Pre: orig i dest són menors que el número de vèrtexs
// Post: Insereix una aresta que connecta des del vèrtex orig cap al vèrtex dest.
        // Si l'aresta ja existia no fa res.
```

Les arestes es guarden en multillistes d'adjacència en memòria dinàmica. Les multillistes estan simplement encadenades i ordenades pel vèrtex origen o destí.
Cal enviar a jutge.org la següent especificació de la classe *graf* i la implementació del mètode dins del mateix fitxer (la resta de mètodes públics ja estan implementats). Indica dins d'un comentari a la capçalera del mètode el seu cost en funció del nombre de vèrtexs *nv* i el nombre d'arestes *na* del graf.

```
#include <vector>
using namespace std;
typedef unsigned int nat;

class graf {
  // Graf dirigit i no etiquetat.
  // Les arestes es guarden en multillistes d'adjacència en memòria dinàmica.
  public:
    // Constructora per defecte. Crea un graf buit de n vèrtexs.
    graf(nat n);

    // Destructora
    ~graf();

    // Retorna un vector amb els successors del vèrtex v
    vector<nat> successors(nat v) const;

    // Retorna un vector amb els predecessors del vèrtex v
    vector<nat> predecessors(nat v) const;

    void insereix (nat orig, nat dest);
    // Pre: orig i dest són menors que el número de vèrtexs
    // Post: Insereix una aresta que connecta des del vèrtex orig cap al vèrtex dest.
          // Si l'aresta ja existia no fa res.

  private:
    nat nv; // Nombre de vèrtexs
    struct node {
      nat orig; // Vèrtex origen
      nat dest; // Vèrtex destí
      node *seg_succ; // Punter al següent successor
```

```
    node *seg_pred; // Punter al següent predecessor
  };
  vector<node *> prim_succ; // Punters al primer successor de cada vèrtex.
                            // La llista de successors està ordenada.
  vector<node *> prim_pred; // Punters al primer predecessor de cada vèrtex.
                            // La llista de predecessors està ordenada.

  // Aquí va l'especificació dels mètodes privats addicionals
};

// Aquí va la implementació del mètode públic insereix i privats addicionals
```

Degut a que jutge.org només permet l'enviament d'un fitxer amb la solució del problema, en el mateix fitxer hi ha d'haver l'especificació de la classe i la implementació del mètode *insereix* (el que normalment estarien separats en els fitxers *.hpp* i *.cpp*).
Per testejar la classe disposes d'un programa principal que crea un graf i després insereix vàries arestes i pregunta pels vèrtexs successors o predecessors d'un vèrtex determinat.

Entrada

L'entrada conté el nombre de vèrtexs del graf seguit de vàries comandes, una per línia, amb el següent format (e, e1 i e2 són naturals):

- insereix e1 e2
- successors e
- predecessors e

Sortida

Per a cada línia d'entrada, escriu una línia amb la comanda d'entrada, el separador ": " i el resultat de la comanda, si en té.
La comanda *insereix* no mostra cap resultat. Les comandes *successors* i *predecessors* envia tots els vèrtexs successors/predecessors d'un donat al canal de sortida separats per espais.

Observació

Només cal enviar la classe requerida i la implementació del mètode *insereix*. Pots ampliar la classe amb mètodes privats. Segueix estrictament la definició de la classe de l'enunciat.
Indica dins d'un comentari a la capçalera del mètode el seu cost en funció del nombre de vèrtexs *nv* i el nombre d'arestes *na* del graf.

Exemple d'entrada 1

```
2
successors 0
successors 1
predecessors 0
predecessors 1
insereix 0 1
successors 0
successors 1
predecessors 0
predecessors 1
insereix 1 0
successors 0
successors 1
predecessors 0
predecessors 1
```

Exemple de sortida 1

```
successors 0:
successors 1:
predecessors 0:
predecessors 1:
insereix 0 1:
successors 0: 1
successors 1:
predecessors 0:
predecessors 1: 0
insereix 1 0:
successors 0: 1
successors 1: 0
predecessors 0: 1
predecessors 1: 0
```

Exemple d'entrada 2

```
3
insereix 0 2
insereix 0 1
insereix 1 2
insereix 0 2
successors 0
successors 1
successors 2
predecessors 0
predecessors 1
predecessors 2
```

Exemple de sortida 2

```
insereix 0 2:
insereix 0 1:
insereix 1 2:
insereix 0 2:
successors 0: 1 2
successors 1: 2
successors 2:
predecessors 0:
predecessors 1: 0
predecessors 2: 0 1
```

Exemple d'entrada 3

```
5
insereix 4 0
insereix 0 2
insereix 0 1
insereix 2 1
insereix 2 4
insereix 2 3
insereix 1 3
insereix 2 1
successors 0
successors 1
successors 2
successors 3
successors 4
predecessors 0
predecessors 1
predecessors 2
predecessors 3
predecessors 4
```

Exemple de sortida 3

```
insereix 4 0:
insereix 0 2:
insereix 0 1:
insereix 2 1:
insereix 2 4:
insereix 2 3:
insereix 1 3:
insereix 2 1:
successors 0: 1 2
successors 1: 3
successors 2: 1 3 4
successors 3:
successors 4: 0
predecessors 0: 4
predecessors 1: 0 2
predecessors 2: 0
predecessors 3: 1 2
predecessors 4: 2
```

Exemple d'entrada 4

```
6
insereix 1 5
insereix 1 0
insereix 3 1
insereix 4 0
insereix 0 5
insereix 5 1
insereix 2 3
insereix 1 0
successors 0
successors 1
successors 2
successors 3
successors 4
```

```
successors 5
predecessors 0
predecessors 1
predecessors 2
predecessors 3
predecessors 4
predecessors 5
```

Exemple de sortida 4

```
insereix 1 5:
insereix 1 0:
insereix 3 1:
insereix 4 0:
insereix 0 5:
insereix 5 1:
insereix 2 3:
insereix 1 0:
successors 0: 5
successors 1: 0 5
successors 2: 3
successors 3: 1
successors 4: 0
successors 5: 1
predecessors 0: 1 4
predecessors 1: 3 5
predecessors 2:
predecessors 3: 2
predecessors 4:
predecessors 5: 0 1
```

Informació del problema

Autoria: Jordi Esteve

Generació: 2026-01-25T17:28:26.008Z