```
/*************************************************************************
Graphe  -  description
-------------------
début                : 2015
copyright            : (C) 2015 par mfallouh_mvirsolvy
*************************************************************************/

//---------- Interface de la classe <Graphe> (fichier Graphe.h) ------
#if ! defined ( GRAPHE_H )
#define GRAPHE_H

//--------------------------------------------------- Interfaces utilisées
#include <string>
#include <map>
#include "Collection.h"
using namespace std;
//------------------------------------------------------------- Constantes

//------------------------------------------------------------------ Types
typedef struct paire {
	// structure associant une cilbe et un referer par le numéro de noeud leur
correspondant

	int NumReferer;
	int NumCible;

	bool const& operator<(const paire & unePaire) const
	// surchage de l'operteur de comparaison pour permettre l'utilisation en
tant que clée dans la map
	//
	// Mode d'emploi :
	// compare d'abord les numReferer entre eux, en cas d'égalité, compare les
numCible
	// Contrat :
	//
	{
		if (this->NumReferer != unePaire.NumReferer)
		{
			return (this->NumReferer < unePaire.NumReferer);
		}
		else
		{
			return (this->NumCible < unePaire.NumCible);
		}
	}
};
//-----------------------------------------------------------------------
// Rôle de la classe <Graphe>
// Cette classe représente un graphe. Elle contient les informations
// nécessaires à sa génération (nœuds et liens). On construit un Graphe
// à partir d'une Collection et des options souhaitées.
//
//-----------------------------------------------------------------------
class Graphe
{
	//------------------------------------------------------------------ PUBLIC

public:
```

```
        //---------------------------------------------------- Méthodes publiques
        static bool EstImage(const string &adresse);
        // Mode d'emploi :
        // vérifie si l'extension est celle d'une image
        // Contrat :
        // le paramètre est une string correspondant à une adresse bien formée

        void GenereFichier(const string & nomFichier);
        // Mode d'emploi :
        // écrit dans(ou crée) le fichier de nom « nomFichier » les instructions
        // de création de graphe, au format spécifié dans l'énoncé.
        //
        // Contrat :
        //


        //------------------------------------------------- Surcharge d'opérateurs
        Graphe & operator = (const Graphe & unGraphe); // déclaré mais non défini
pour empêcher son utilisation
        // Mode d'emploi :
        //
        // Contrat :
        //


        //-------------------------------------------- Constructeurs - destructeur
        Graphe(const Graphe & unGraphe); // déclaré mais non défini pour empêcher
son utilisation
        // Mode d'emploi (constructeur de copie) :
        //
        // Contrat :
        //

        Graphe(const Collection &aCol, const bool e = false, const int t = -1);
        // Mode d'emploi :
        // parcourt la Collection entrée en paramètre afin de remplir les
dictionnaires.
        // Prend en compte des options e et t.
        // Contrat :
        //

        virtual ~Graphe();
        // Mode d'emploi :
        //
        // Contrat :
        //

        //------------------------------------------------------------------ PRIVE

protected:
        //----------------------------------------------------- Méthodes protégées

private:
        //-------------------------------------------------------- Méthodes privées

protected:
        //----------------------------------------------------- Attributs protégés
```

```
        map<string, int > noeuds;  // dictionnaire de nœuds associant à chaque
                                                    // adresse de page son numero
pour le tracé du graphe

        map<paire, int> liens;            //dictionnaire de liens, associant à
chaque paire(structure :
                                                      //referer et cible,
identifiés par leurs numéros de noeud) son nombre de hits.

        int valeurNoeud;                  // la valeur du prochain noeud ajouté dans
la la map noeud
private:
        //------------------------------------------------------ Attributs privés

        //------------------------------------------------------------ Classes amies

        //------------------------------------------------------------- Classes privées
        void creeGrapheHeure(map<string, Cible > ::const_iterator &Cible, const
size_t &heure, bool e);
        // Mode d'emploi :
        // Pour la cible entré en paramètre , et l'heure, parcours les logs en
mettant à jour les attributs liens et noeuds.
        // Contrat :
        // heure < 24

        bool creeNoeud(const string &page, const int &valeurNoeud);
        // Mode d'emploi :
        // Crée le noeud, si jamais il existe déjà, alors rien ne se passe
        // Contrat :
        //
        //-------------------------------------------------------------- Types privés

};

//---------------------------------------- Types dépendants de <Graphe>

#endif // GRAPHE_H
```