TP3. Mise en oeuvre d'une application de la classe générique Graphe<S,T>. Carte routière simplifiée (2h)

Introduction

Pour vérifier que notre définition de graphe est fonctionnelle, nous choisissons un exemple d'application : une carte routière simplifiée.
Ceci nous amène à définir les classes suivantes :

- classe *InfoSommetCarte* : contient les informations relatives à un lieu d'une carte routière

- classe *InfoAreteCarte* : contient les informations relatives à une arête. Cette classe permet la définition de la classe non générique ***Arete<InfoAreteCarte, InfoSommetCarte>*** qui représente une route d'une carte routière.

- classes *DessinGraphe* et *DessinGrapheRecuitSimule*: elles permettent de dessiner une carte routière par l'intermédiaire d'un fichier texte à l'aide de l'appli JAVA *bsplines*.

Une fois ces 5 classes écrites, nous définissons une fonction *main*() qui construit une petite carte routière, l'affiche en mode console puis la dessine (par l'intermédiaire de l'appli de dessin JAVA *bsplines*). La carte routière est définie comme instance de la classe non générique ***Graphe <InfoAreteCarte, InfoSommetCarte>***.

1. Classe InfoSommetCarte

Cette classe représente les informations relatives à un lieu d'une carte routière. Pour simplifier, un lieu est défini par un nom et une position.
La classe *InfoSommetCarte* contient donc les 2 attributs suivants :
nom : de type *string*
position : de type *Vecteur2D* (cette classe est fournie)

Ecrire la classe *InfoSommetCarte* ainsi définie. Pour simplifier, *nom* et *position* peuvent être publics.
Munir *InfoSommetCarte* d'un constructeur, d'un opérateur de conversion en *string* et de l'opérateur << d'écriture sur un flux.
Il est inutile d'écrire destructeur, constructeur de copie ou getters et setters.

2. Classe InfoAreteCarte

La classe *InfoAreteCarte* contient l'information associée à une arête d'une carte routière. Pour simplifier, seul le coût de l'arête est noté. Ce coût est supposé être un nombre réel positif ou nul.
La classe *InfoAreteCarte* contient donc un unique attribut, noté *cout*, de type *double*.
Ecrire la classe *InfoAreteCarte* ainsi définie. Pour simplifier, l'attribut *cout* peut être public.
Munir *InfoAreteCarte* d'un constructeur, d'un opérateur de conversion en *string* et de l'opérateur << d'écriture sur un flux.
Il est inutile d'écrire destructeur, constructeur de copie ou getters et setters.

3. Classe DessinGrapheRecuitSimule

Elle permet de créer un fichier texte contenant des instructions pour dessiner la carte routière. Le graphe peut ensuite être dessiné grâce à l'appli JAVA *bsplines* qui exploite les données du fichier texte. La classe *DessinGrapheRecuitSimule* est fournie.
La classe *DessinGrapheRecuitSimule* ne contient que des méthodes statiques (il est inutile de l'instancier) dont la principale est :

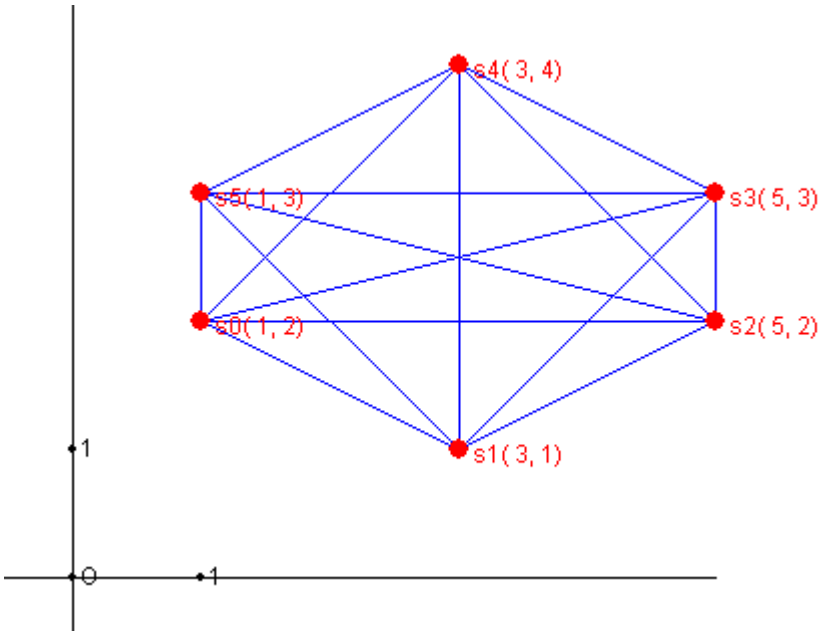
```
/**
écrit le graphe-carte routière sur le fichier texte de dessin of.
suppose of déjà ouvert en écriture seule

rayonSommet est le rayon des sommets sur l'écran
*/
static void ecritGraphe(ofstream & of,
Graphe<InfoAreteCarte,InfoSommetCarte> & graphe,
                                        const Vecteur2D & coinBG,
                                        const Vecteur2D & coinHD,
                                        const string & couleurRepere,
                                        const double & rayonSommet,
                                        const string & couleurSommets,
                                        const string & couleurAretes);
```

où :
of est le fichier texte résultat. Il est supposé déjà ouvert en écriture à l'appel.
graphe est la carte routière à dessiner.
coinBG (pour coin bas gauche) et *coinHD* (pour coin haut droit) sont les coins du rectangle à visualiser en coordonnées du monde.
rayonSommet est le rayon des sommets en coordonnées écran.
Les couleurs peuvent être choisies parmi "black", "blue", "red", "green", "yellow" et "cyan".

4. fichier TestGraphePourRecuitSimule.cpp et fonction main()

Le fichier *TestGraphePourRecuitSimule.cpp* contient la fonction *main()* chargée de vérifier les fonctionnalités de la classe *Graphe<InfoAreteCarte, InfoSommetCarte>*.

Il est demandé de créer la carte routière suivante :



Cette carte est créée puis affichée en mode console. La carte est ensuite dessinée.
Pour le calcul de la distance entre deux points *A* et *B* (pour renseigner le coût des arêtes), on peut utiliser la formule suivante : *AB* = | ***OB*** **-** ***OA***|.
OB et ***OA*** étant les vecteurs position de *A* et *B*.
La soustraction et la fonction norme euclidienne (fonction *norme*(...) non membre) d'un vecteur sont définies dans *AlgebreLineaire.h* (fichier fourni).
Il est utile de définir une classe *OutilsCarteRecuitSimule* rassemblant les méthodes utiles à la classe *Graphe<InfoAreteCarte, InfoSommetCarte>*.
Il est alors très pratique de définir dans *OutilsCarteRecuitSimule* les deux méthodes statiques suivantes :

- ```
  static double distance( const Sommet<InfoSommetCarte> * s1,
                          const Sommet<InfoSommetCarte> *
  s2);
  ```

  qui calcule la distance euclidienne entre deux lieux *s1* et *s2*.

- ```
  static Arete<InfoAreteCarte, InfoSommetCarte> *
  creeArete(Sommet<InfoSommetCarte> * sA,
  Sommet<InfoSommetCarte> * sB, Graphe<InfoAreteCarte,
  InfoSommetCarte> & graphe);
  ```

qui crée dans *graphe* l'arête *sA* - *sB* (et qui exploite la méthode précedente).

Annexe

1. Classe DessinGraphe

Cette classe est utilisée par la classe *DessinGrapheSimule*.

```
#pragma once
#include <fstream>
#include "Vecteur2D.h"
#include "Graphe.h"
#include "InfoAreteCarte.h"
#include "InfoSommet.h"

using namespace std;
/**

méthodes nécessaires pour écrire un graphe-carte routière dans un fichier
texte de dessin avec visualisation d'un chemin

On suppose toujours que of est déjà ouvert en écriture seule
*/
class DessinGraphe
{
public:

static void ecrireEntete(ofstream & of, const string & titre, const string &
legende, const string & resume, const Vecteur2D & coinBG, const
Vecteur2D & coinHD);
static void ecritNombrePointsRemarquables(ofstream & of, int
nombrePoints);
static void ecritRepere(ofstream & of, const string & couleur);

static void ecritNombreCourbes(ofstream & of, int nombreCourbes);
};
```

----------DessinGraphe.cpp ----------

```
#include "DessinGraphe.h"


/**

méthodes nécessaires pour écrire un graphe-carte routière dans un fichier
texte de dessin avec visualisation d'un chemin

On suppose toujours que of est déjà ouvert en écriture seule
*/
//class DessinGraphe
//{
//public:
```

```
/*static*/ void DessinGraphe::ecrireEntete(ofstream & of, const string &
titre, const string & legende, const string & resume, const Vecteur2D &
coinBG, const Vecteur2D & coinHD)
{
of << "titre = " << titre << endl;
of << "legende = " << legende << endl;
of << "resume = " << resume << endl;
of << "type de scene = courbes" << endl;
of << "coin bas gauche de la figure sur l'écran en coordonnées monde = "
<< coinBG << endl;
of << "coin haut droit de la figure sur l'écran en coordonnées monde = "
<< coinHD << endl;
}

/*static*/ void DessinGraphe::ecritNombrePointsRemarquables(ofstream &
of, int nombrePoints)
{
of << "nombre de points remarquables = " << nombrePoints << endl;
}

/*static*/ void DessinGraphe::ecritRepere(ofstream & of, const string &
couleur)
{
of << "point remarquable = 2 black (0,0) O" << endl;
of << "point remarquable = 2 black (1,0) 1" << endl;
of << "point remarquable = 2 black (0,1) 1" << endl;
}

/*static*/ void DessinGraphe::ecritNombreCourbes(ofstream & of, int
nombreCourbes)
{
of << "nombre de courbes = "<< nombreCourbes << endl;
}
```

2. Classe DessinGrapheRecuitSimule

```
#pragma once

#include <fstream>
#include <string>

#include "Graphe.h"
#include "InfoSommetCarte.h"
#include "InfoAreteCarte.h"

using namespace std;
/**
définit les outils nécessaires pour dessiner un graphe représentant une
carte routière.
```

```
But : Application de l'algorithme du recuit simulé à la recherche du
problème du voyageur de commerce

cette classe définit les outils qui ne sont pas déjà dans DessinGraphe (cf.
classe DessinGraphe)
*/
class DessinGrapheRecuitSimule
{
public:
/**

rayonSommet est le rayon des sommets sur l'écran
*/
static void ecritSommets(ofstream & of, const Graphe<InfoAreteCarte,
InfoSommetCarte> & graphe, const double & rayonSommet, const string
& couleur);

static void ecritAretes(ofstream & of, const Graphe<InfoAreteCarte,
InfoSommetCarte> & graphe, const string & couleur);


/**
écrit le graphe-carte routière sur le fichier texte de dessin of.
suppose of déjà ouvert en écriture seule

rayonSommet est le rayon des sommets sur l'écran
*/
static void ecritGraphe(ofstream & of,
Graphe<InfoAreteCarte,InfoSommetCarte> & graphe,
                                        const Vecteur2D & coinBG, const
Vecteur2D & coinHD,
                                        const string & couleurRepere,
                                        const double & rayonSommet,
                                        const string & couleurSommets,
                                        const string & couleurAretes);
};
```

---------- DessinGrapheRecuitSimule.cpp ----------

```
#include "DessinGraphe.h"
#include "DessinGrapheRecuitSimule.h"

/**

rayonSommet est le rayon des sommets sur l'écran
*/
```

```
/*static*/ void DessinGrapheRecuitSimule::ecritSommets( ofstream & of,
const Graphe<InfoAreteCarte, InfoSommetCarte> & graphe,

                const double & rayonSommet, const string & couleur)
{
PElement<Sommet<InfoSommetCarte>> * l;

for ( l = graphe.lSommets; l; l = l->s)
        {
        InfoSommetCarte * info = &(l->v->v);
        of << "point remarquable = "<< rayonSommet<< " "<<couleur<<"
"<< info->position<<" " <<info->nom << info->position << endl;
        }


}

/*static*/ void DessinGrapheRecuitSimule::ecritAretes(ofstream & of, const
Graphe<InfoAreteCarte, InfoSommetCarte> & graphe, const string &
couleur)
{
PElement<Arete<InfoAreteCarte,InfoSommetCarte>> * l;

for ( l = graphe.lAretes; l; l = l->s)
        {
        of << "couleur =  " << couleur<<endl;
        of << "nombre de points = 2" << endl;
        of << l->v->debut->v.position << endl;
        of << l->v->fin->v.position << endl;
        }
}


/**
écrit le graphe-carte routière sur le fichier texte de dessin of.
suppose of déjà ouvert en écriture seule

rayonSommet est le rayon des sommets sur l'écran
*/
/*static*/ void DessinGrapheRecuitSimule::ecritGraphe(ofstream & of,
Graphe<InfoAreteCarte,InfoSommetCarte> & graphe,
                                        const Vecteur2D & coinBG, const
Vecteur2D & coinHD,
                                        const string & couleurRepere,
                                        const double & rayonSommet,
                                        const string & couleurSommets,
                                        const string & couleurAretes)
{
string resume;
```

```
resume = "carte routière";

DessinGraphe::ecrireEntete( of,  "carte routière",  "carte routière simplifiée
représentée par un graphe", resume,  coinBG,  coinHD);

DessinGraphe::ecritNombrePointsRemarquables( of,
3+graphe.nombreSommets());
DessinGraphe::ecritRepere( of, couleurRepere);
ecritSommets( of,  graphe, rayonSommet, couleurSommets);

DessinGraphe::ecritNombreCourbes( of, graphe.nombreAretes());
ecritAretes( of,  graphe, couleurAretes);
}
```