Types abstraits

TYPE ABSTRAIT Visee

Ce type abstrait définit une Visee

Opérations de base :
Constructeur Visee : creerViseeVide() : Visee
Constructeur Visee : creerVisee(Entier numDepart, Entier numArrivee, Reel longueur, Reel angle) : Visee

Transformateur Visee : setNumDepart(Entier numDepart) : Visee
Transformateur Visee : setNumArrivee(Entier numArrivee) : Visee
Transformateur Visee : setLongueur(Reel longueur) : Visee
Transformateur Visee : setAngle(Reel angle) : Visee

Observateur Visee : getNumDepart() : Entier
Observateur Visee : getNumArrivee() : Entier
Observateur Visee : getLongueur() : Vecteur
Observateur Visee : getAngle() : Vecteur

Axiomes :
getNumDepart(setNumDepart(visee,numDepart))=numDepart
getNumArrivee(setNumArrivee(visee,numArrivee))=numArrivee
getLongueur(setLongueur(visee,longueur))=longueur
getAngle(setAngle(visee,angle))=angle

TYPE ABSTRAIT Topographie

Ce type abstrait définit une Topographie

Opération de base
Constructeur Topographie : creerTopographie () : Topographie
Constructeur Topographie : creerTopographie (Chaine nom, Point pointEntree, Station entree, Chaine date, Chaine auteurs, Entier unite, Reel precLongueur, Reel precAngle, Liste listeSeries, Map positions) : Topographie

Transformateur Topographie : setNom (Chaine nom) : Topographie
Transformateur Topographie : setPointEntree (Point point) : Topographie
Transformateur Topographie : setStationEntree (Station entree) : Topographie
Transformateur Topographie : setDate(Chaine date) : Topographie
Transformateur Topographie : setAuteurs (Chaine auteurs) : Topographie
Transformateur Topographie : setUnite (Entier unite) : Topographie
Transformateur Topographie : setPrecLongueur(Reel precLongueur) : Topographie

Transformateur Topographie : setPrecAngle (Reel precAngle) : Topographie
Transformateur Topographie : setListeSeries(Liste listeSeries) : Topographie
Transformateur Topographie : setPositions(Map positions) : Topographie

Observateur Topographie : getNom() : Chaine
Observateur Topographie : getPointEntree () : Point
Observateur Topographie : getStationEntree () : Station
Observateur Topographie : getDate () : Chaine
Observateur Topographie : getAuteurs() : Chaine
Observateur Topographie : getUnite() : Entier
Observateur Topographie : getPrecLongueur () : Reel
Observateur Topographie : getPrecAngle () : Reel
Observateur Topographie : getListeSeries () : Liste
Observateur Topographie : getPositions () : Map
Observateur Topographie : getNbSeries() : Entier

Axiomes
getNom(setNom (topographie,nom))=nom
getStationEntree(setStationEntree(topographie,entree))=entree
getDate(setDate (topographie,date))=date
getAuteurs(setAuteurs (topographie, auteurs))= auteurs
getUnite(setUnite (topographie, unite))=unite
getPrecLongueur (setPrecLongueur (topographie, precLongueur))= precLongueur
getPrecAngle(setPrecAngle (topographie,precAngle))=precAngle
getListeSeries (setListeSeries (topographie, listeSeries))=listeSeries
getPositions (setPositions (topographie, positions))= positions

Pré-conditions :

TYPE ABSTRAIT Station

Ce type abstrait définit une Station

Opérations de base
Constructeur Station : creerStation () : Station
Constructeur Station : creerStation (Entier numero, Reel longueur, Reel angle, Reel x, Reel y) : Station

Transformateur Station : setNumero (Entier numero) : Station
Transformateur Station : setLongueur (Reel longueur) : Station
Transformateur Station : setAngle (Reel angle) : Station
Transformateur Station : setX(Reel x) : Station
Transformateur Station : setY(Reel y) : Station

Observateur Station : getNumero() : Entier
Observateur Station : getLongueur() : Reel
Observateur Station : getAngle() : Reel

Observateur Station : getX() : Reel
Observateur Station : getY() : Reel
Observateur Station : estVisitee() : Booleen

Axiomes :
getNumero(setNumero(station,numero))=numero
getLongueur(setLongueur(station,longueur))=longueur
getAngle(setAngle(station,angle))=angle
getLongueur(setX(station,x))=x
getLongueur(setY(station,y))=y
NON estVisitee(creerStration())
NON estVisitee(creerStation(numero, longueur, angle))

Pré-conditions :
definie(creerStation(numero, longueur, angle)) ⇔ (longueur >= 0) et (angle >= 0) et (angle <= 400)
definie(setAngle(angle)) ⇔ (angle >= 0) et (angle <= 400)
definie(setLongueur(longueur)) ⇔ longueur >=0

TYPE ABSTRAIT Serie

Ce type abstrait définit une Série

Opérations de base
Constructeur Serie : creerSerie () : Serie
Constructeur Serie : creerSerie (Chaine nom, Entier numero, station depart, station arrivee, liste listeStation) : Serie
Constructeur Serie : selectSerie(Topographie T,Entier i) : Série

Transformateur Serie : setNom (Chaine nom) : Serie
Transformateur Serie : setNumero (Entier numero) : Serie
Transformateur Serie : setStationDepart (station depart) : Serie
Transformateur Serie : setStationArrivee (station arrivee) : Serie
Transformateur Serie : setListeStation (liste listeStation) : Serie

Observateur Serie : getNom () : Chaine
Observateur Serie : getNumero () : Entier
Observateur Serie : getStationDepart () : station
Observateur Serie : getStationArrivee () : station
Observateur Serie : getListeStation () : liste
Observateur Serie : getNbStations() : Entier

Axiomes :
creerSerie(nom,
numero,depart,arrivee,listeStation)=setNom(setNumero(setStationDepart(setStationArrivee(setListeStation(creerStation(),liste),arrivee),depart),numero),nom)

getNom(setNom(serie,nom))=nom
getNumero(setNumero(serie,numero))=numero
getStationDepart(setStationDepart(serie,depart))=depart
getStationArrivee(setStationArrivee(station,arrivee))=arrivee
getListeStation(setListeStation(serie,liste))=liste

TYPE ABSTRAIT Matrice

Ce type abstrait définit l'objet mathématique Matrice

Opérations de base :
Constructeur Matrice : creerMatriceVide() : Matrice
Constructeur Matrice : creerMatrice(Entier nbLignes, Entier nbColonnes, Vecteur tab) : Matrice

Transformateur Matrice : setNbLignes(Entier nbLignes) : Matrice
Transformateur Matrice : setNbColonnes(Entier nbColonnes) : Matrice
Transformateur Matrice : setValeurs(Entier i, Entier j, Reel valeur) : Matrice

Observateur Matrice : getNbLignes() : Entier
Observateur Matrice : getNbColonnes() : Entier
Observateur Matrice : getValeur(Entier i, Entier j) : Vecteur

Axiomes :
creerMatrice(nbLignes, nbColonnes, tab) =
setNbLignes(setNbColonnes(setValeurs(creerMatriceVide(),tab),nbColonnes),nbLignes)
getNbLignes(setNbLignes(mat,nbLignes))=nbLignes
getNbColonnes(setNbColonnes(mat,nbColonnes))=nbColonnes
getValeurs(setValeurs(mat,tab))=tab

Type Abstrait : Chaine

Concept :
Ce type définit une chaine de caractère
Fin du concept

Opérations de base :
Constructeur Chaine : creerChaineVide() : Chaine
Constructeur Chaine : concatener(Chaine a, Chaine b) : Chaine
Constructeur Chaine : concatener(Chaine a, caractere c) : Chaine

Observateur Chaine : toReel() : Reel
Observateur Chaine : recCarac (Entier numeroCaractere) : caractere
Observateur Chaine : taille() : Entier

Axiomes :
longueur(chaineVide) = 0
longueur(concatener(ch1, ch2)) = longueur(ch1) + longueur(ch2)
longueur(caractere(c)) = 1
recCarac(ch,caractere(c),1) = c
i <= longueur(ch1) Þ recCarac(concatener(ch1,ch2), i) = recCarac(ch1,i)
i > longueur(ch1) Þ recCarac(concatener(ch1,ch2), i) = recCarac(ch2,i - longueur(ch1))

TYPE ABSTRAIT caractere

Début du concept
ce type abstrait définit un caractère
Fin du concept

opérateur de base
Constructeur caractere : creerEspace() : caractere
Constructeur caractere : creera() : caractere
Constructeur caractere : creerA() : caractere
Constructeur caractere : carSuivant(caractere c) : caractere
Constructeur caractere : creerRetourLigne() : caractere

Observateur caractere : carCode() : Entier
Observateur caractere : estRetourLigne() : Booleen
Observateur caractere : estTabulation() : Booleen
Observateur caractere : estEspace() : Booleen
fin d'opération de base

Axiomes :
estEgal(carCode(carSuivant(c)),carCode(c) + 1)
estEspace(creerEspace())
estRetourLigne(estRetourLigne())

ALGORITHMES

Question 1 :

Constructeur : chargerTopographie(Chaine nomfichier) : Topographie

References locales :
Topographie topo
Chaine s
Flot flot
Entier numserie
Entier numstation
Entier c1
Entier c2
Point point
Chaine date
Chaine auteurs
Reel unite
Reel precangle
Reel preclongueur
Entier iserie
Entier istation
Chaine nomseriecourante
Entier numeroStationDep
Entier numeroStationArr
Entier numeroSerieDep
Entier numeroSerieArr
Serie scourante
Station stationcourante
Reel longueurStationCourante
Reel angleStationCourante
Liste listeStations

DEBUT

Numserie ← 0
iserie ← 0
istation ← 0
map ← creerMap()

//Création du flot
Flot ← creerFlot(s)
//Ouverture du flot
Ouvrir(flot,nomfichier)

//Gestion de l'erreur d'ouverture du flot
SI (estEgal(flot,RefNull)) ALORS
DEBUT
    Afficher ("Erreur lors de l'ouverture du flot")
    Retourner refNulle
FIN

//Si tout c'est bien passé on peut commencer la recuperation

```
TANT QUE (NON fdf(flot)) FAIRE
DEBUT
        topo ← creerTopographie()

        c1 ← consulter(prendre(flot))
        c2 ← consulter(prendre(flot))
        SI (estEgal(c1,-5) et estEgal(c2,1)) ALORS
        DEBUT
                s ←- consulter(prendre(flot))
                setNom(topo, chaine)
        FIN
        SINON
        DEBUT
                Afficher ("Erreur de syntaxe dans le fichier")
                Retourner refNulle
        FIN

        c1 ← consulter(prendre(flot))
        c2 ← consulter(prendre(flot))
        SI (estEgal(c1,-4) et estEgal(c2, 1) ALORS
        DEBUT
                valX ← consulter(prendre(flot))
                valY ← consulter(prendre(flot))
                setX(point,valX)
                setY(point,valY)
                setPointEntree(topo, p)
        FIN
SINON
        DEBUT
                Retourner refNulle
        FIN

        c1 ← consulter(prendre(flot))
        c2 ← consulter(prendre(flot))
        SI (estEgal(c1,-2) et estEgal(c2, 1) ALORS
        DEBUT
                date ← consulter(prendre(flot))
                auteurs ← consulter(prendre(flot))
                setDate(topo, date)
                setAuteurs(topo, auteurs)
        FIN
SINON
        DEBUT
                Retourner refNulle
        FIN

        c1 ← consulter(prendre(flot))
        c2 ← consulter(prendre(flot))
        SI (estEgal(c1,-1) et estEgal(c2, 1)) ALORS
```

```
                unite ← consulter(prendre(flot))
                precangle ← consulter(prendre(flot))
                preclongueur ← consulter(prendre(flot))
                setUnite(topo, unite)
                setPrecAngle(topo, precangle)
                setPrecLongueur(topo, preclongueur)
        FIN
SINON
        DEBUT
                Retourner refNulle
        FIN

        //On a recupéré les infos de la topographie, maintenant on va recupérer les infos sur les séries et les stations

        TANT QUE (NON fdf(flot)) FAIRE
        DEBUT
                iserie ← iserie + 1
                numserie ← consulter(prendre(flot))
                c2 ← consulter(prendre(flot))
                SI (estEgal(numserie,iserie) ET estEgal(c2,-2)) ALORS
                DEBUT
                        Nomseriecourante ← consulter(prendre(flot))
                        Ajouter(map,numserie,nomseriecourante)
                FIN
        SINON
                DEBUT
                        Retourner refNulle
                FIN


                SI (estEgal(numserie,iserie) ET estEgal(c2,-1)) ALORS
                DEBUT
                        numeroSerieDep ←- ← consulter(prendre(flot))
                    numeroStationDep ← consulter(prendre(flot))
                    numeroSerieArr ← consulter(prendre(flot))
                        numeroStationArr ← consulter(prendre(flot))
                        scourante ← creerSerie()
                        setNumero(scourante, iserie)

                        TANT QUE (estEgal(numserie,iserie))FAIRE
                        DEBUT
                                istation ← istation + 1
                                numserie ←- ← consulter(prendre(flot))
                                numstation ←- ← consulter(prendre(flot))

                SI (estEgal(numserie,iserie)) ALORS
                                DEBUT
                                longueurStationCourante ← consulter(prendre(flot))
```

```
                    angleStationCourante ← consulter(prendre(flot))
                    stationcourante ←
creerStation(numstation,longueurStationCourante,angleStationCourante)
                    ajouter(listeStations, stationcourante)

        SI (estEgal(numserie , numeroSerieDep) ET estEgal(numstation , numeroStationDep)) ALORS
                    DEBUT
                    setStationDepart(scourante, stationcourante)
                    FIN
                    SINON DEBUT
                SI (estEgal(istation , 1) OU estEgal(numstation , 0)) ALORS
                            DEBUT
                            setStationDepart(scourante, getStation(numeroSerieDep,
numeroStationDep))
                            FIN
                            SINON
                            DEBUT
                                    Retourner refNulle
                            FIN
                    FIN
                    FIN
            FIN
            SINON
                    DEBUT
                    Retourner refNulle
                    FIN

            setListeStation(scourante, listeStations)

            SI (estEgal(numserie , numeroSerieArr) ET estEgal(numstation ,
numeroStationArr)) ALORS                                    DEBUT
                    setStationArrivee(scourante, stationcourante)
            FIN
        SINON
            DEBUT
                    setStationArrivee(scourante, getStation(numeroSerieArr,
numeroStationArr))
            FIN

            ajouter(listeSeries, scourante)
            ajouter(map, iserie, nomseriecourante)
            setListeStations(scourante, listeStations)
          FIN
FIN

setListeSeries(topo, listeSeries)
setMap(topo, map)

fermer(flot, nomfichier)
```

```
SI (estEgal(flot,RefNull)) ALORS
DEBUT
        Afficher ("Probleme lors de la fermeture du fichier")
        Retourner refNull
FIN

Retourner topo

FIN
```

Question 2 :

Transformateur : Station modifierStation(Reel longueur, Reel angle) : Station

Transformé station

DEBUT

    setLongueur(station,longueur)

    setAngle(station,angle)

    retourner(station)

FIN

Question 3 :

```
Transformateur : Topographie supprimerSerie(Entier numero) : Topographie
Transforme topo
Références locales :
Liste l, ltemp
Entier i, j
DEBUT
        l ← getListeSeries(topo)
        i ← 0
        j ← 0

        TANTQUE i < longueur(l) FAIRE
        DEBUT
        SI estEgal(numero, getNumero(getElement(l,i)))
DEBUT
        TANTQUE j<i FAIRE
        DEBUT
        Ajouter(ltemp,premier(l))
        Supprimer(l)
        J ← j+1
        FIN
        Supprimer(l)
        TANTQUE NON estVide(l) FAIRE
        DEBUT
        Ajouter(ltemp,premier(l))
        Supprimer(l)
        FIN
FIN
        I ←i+1
        FIN

        setListeSeries(topo, ltemp)
        retourner topo
FIN
```

Question 4 :

Observateur : Topographie recupererInfos(Chaine nom) : Vecteur
Observé topo
Références locales :
Liste l
Map map
Vecteur res
Serie s
DEBUT
	TANTQUE i < longueur(l) FAIRE
	DEBUT
	SI estEgal(nom, getNom(getElement(l,i)))
DEBUT
	S ← getElement(l,i)
	Ajouter(res, getNumero(s))
	Ajouter(res, getNom(s))
	Ajouter(res,getStationDepart(s))
	Ajouter(res,getStationArrivee(s))
	Ajouter(res,getListeStations(s))
FIN
Retourner res
FIN
	Retourner RefNull
FIN

Question 5 :

Transformateur topographie : ajouterSerie(Chaine nom, Entier numeroSerie, Entier numeroSerieDepart, Entier numeroStationDepart, Entier numeroSerieArrivee, Entier numeroStationArrivee) : Topographie
Transformé topo
References locales :
Serie serie
Sortie sortie
Entree entree
Station station
Caractere char
Reel v1, v2
Liste listeStation

DEBUT

Serie ← creerSerie()
setNumero(serie, numeroSerie)
setNom(serie, nom)

listeStation ← creerListe()

SI estEgal(numeroSerie, numeroSerieDepart) ALORS
DEBUT
	Station ←creerStation()
	setStationDepart(serie,station)
	ajouter(listeStation,station)
FIN
SINON
DEBUT
	Station ← getStation(numeroSerie,numeroStationDepart)
	setStationDepart(serie, station)
	s ← creerStation()
FIN

SI non estEgal(numeroSerie,numeroStationArrivee) ALORS
DEBUT
	Station ← getStation(numeroSerie,numeroSerie, numeroStationArrivee)
	setStationArrivee(serie, station)
FIN

SI estEgal(numeroSerie, numeroStationArrivee) ALORS
DEBUT
	Station ← premier(listeStation)
	setStationArrivee(serie,station)

FIN
FIN

Question 6 :

Observateur : Topographie sauvegarderTopographie(Chaine nomFichier) : Fichier

Observé topo

Références locales :

Liste listeSeries

Chaine string

Sortie sortie

Entier i, k

Serie s

Liste l, l1

Entier min

DEBUT

i ← 1

sortie =creerSortie(nomFichier)

ecrire(-5 1)

ecrire(getNom(topo))

sautLigne()

ecrire(-4 1)

ecrire(getX(getStationEntree(lab)) recValY(getStationEntree(topo)))

sautLigne()

ecrire(-3 1)

sautLigne()

ecrire(-2 1)

ecrire(getAuteurs(topo))

sautLigne()

ecrire(-1 1)

```
ecrire(getUnite(topo) getPrecLongueur(topo) getPrecAngle(topo))
sautLigne()
TANT QUE  i <= longueur(getListeSerie(topo)) FAIRE
        DEBUT
        l ← getListeSeries(topo)
        TANT QUE (non estVide(l)) FAIRE
                DEBUT
                SI ( estEgal(getNumero(premier(l)),i) ) ALORS
                        DEBUT
                        s ← premier(l)
                        FIN
                FIN
        ecrire(i -2 )
        ecrire(getNom(s))
        sautLigne()
        ecrire(i -1 )
        SI (appartientAutreSerie(getStationDepart(s), topo) ALORS
                DEBUT
                m ← minNumeroSeries(topo, getStationDepart(s))
                ecrire(m getNumero(getStationDepart(s)))
                FIN
        SINON
                DEBUT
                ecrire(getNumero(s) getNumero(getStationDepart(s)))
                FIN

        SI (appartientAutreSerie(topo, getStationArrivee(s)) ALORS
                DEBUT
                m ← minNumeroSeries(topo, getStationArrivee(s))
                ecrire(m getNumero(getStationArrivee(s)))
                FIN
        SINON
                DEBUT
                ecrire(getNumero(s) getNumero(getStationArrivee(s)))
```

```
            FIN
    sautLigne()

    k ← 0
    l1 ← inverserListe(recListeStation(s))
    TANT QUE (non estVide(l1)) FAIRE
            DEBUT
            ecrire(i k )
            ecrire(getLongueur(premier(l1)) getAngle(premier(l1)))
            l1 ← reste(l1)
            sautLigne()
            k ← k+1
            FIN
    i ← i+1
    FIN
```

Observateur : Topographie appartientAutreSerie(Station station) : BOOLEEN

Observé topo

References locales :

Liste listeserie

Liste listestation

Serie seriecourante

Station stationcourante

DEBUT

Listeserie ← getListeSeries(topo)

TANTQUE NON estVide(listeserie) FAIRE
DEBUT
    seriecourante ← premier(listeSerie)
    Listestation ← getListeStation(seriecourante)

```
listeSerie ← reste(listeSerie)

TANTQUE NON estVide(listestation) FAIRE
DEBUT
Stationcourante ← premier(listestation)
Listestation ← reste(listestation)

SI estEgal(getNumero(station),getNumero(stationcourante)) ALORS
DEBUT
        Retourner vrai
FIN

        FIN

FIN

Retourner faux


FIN



Transformateur  Liste : inverserListe() : Liste
Transformé liste


Référence local :
Liste listetemp



DEBUT

    Listetemp <- listeVide()



        TANTQUE(NON estVide(liste)) FAIRE

        DEBUT

      ajouter(listetemp,premier(liste))
      supprimer(liste)
```

FIN

Retourner (listetemp)

FIN

Question 7 :

```
Transformateur Topographie : AffectationNumeroUnique() : Topographie
Transformé topo
Références locales
Entier nbse, i, j, nombre, nbexistant
Vecteur bout
Serie se, se2
Station st, st2

DEBUT
//Tout d'abord on attribue les numéros de stations à la série 1, du fait que la première station de
//cette série est l'entrée, donc le point 0
        se ← selectSerie(topo,1)
        nombre ← 0
        TANTQUE non(estVide(getListeStation(se))) Faire
        DEBUT
                st ← selectStation(se,nombre+1)
                st ← setNumero(st, nombre)
                nombre ← nombre + 1
                se ← reste(se)
        FIN
//Ensuite on remplit le reste. Boucle pour numéroter toutes les séries suivantes
                nbse ← getNbSeries(topo)
                i ← 2
                TANTQUE i <=nbse FAIRE
                DEBUT
                        se ← selectSerie(topo,i)
//Si la série commence d'un point existant, on attribue à la première station son numéro déjà //existant
```

```
                        Si departExistant(se) Alors
                        DEBUT
                                bout ← getStationDepart (se)
                                se2 ← selectSerie(topo,recVal(bout,1))
                                st2 ← selectStation(se2,recVal(bout,2))
                                nbexistant ← getNum(st2)
                                st ← selectStation(se,1)
                                setNumero(st, nbexistant)
                        FIN
//Si la série ne commence pas d'un point existant alors on lui attribue un nouveau numéro
                        Sinon
                                st ← selectStation(se,1)
                                setNumero(st, nombre)
                                nombre ← nombre + 1
                        FIN
                        I← i+1
                FIN
//Ensuite on remplit les stations suivante, sauf la dernière (pour le cas où la station d'arrivée existe
//déjà)

                        J ← 2
                        TANTQUE j <= getNbStations(se) -1 FAIRE
                        DEBUT
                                st ← selectStation(se,j)
                                st ← setNumero(st, nombre)
                                nombre ← nombre + 1
                                se ← reste(se)
                                j ← j+1
                        FIN
//Numérotation de la dernière station
                        Si arriveeExistante(se) Alors
                        DEBUT
                                bout ← getStationArrivee(se)
                                se2 ← selectSerie(topo,recVal(bout,1))
                                st2 ← selectStation(se2,recVal(bout,2))
                                nbexistant ← getNumero(st2)
                                st ← selectStation(se,1)
                                setNumero(st, nbexistant)
                        FIN
//Si la série ne FINit pas par un point existant alors on lui attribue un nouveau numéro
                        Sinon
                                st ← selectStation(se,1)
                                setNumero(st, nombre)
                                nombre ← nombre + 1
                        FIN
                FIN
                Retourner topo
        FIN
```

```
//Fonction retournant le nombre de stations qu'il y a au total dans la topographie
//Il suffit de retourner le numéro de la station le plus grand de la dernière série, et c'est forcément l'avant-dernière station
//(dans le cas où la station d'arrivée existe déjà) ou la dernière (dans le cas contraire)
Observateur Topographie : getTotalStations() : Entier
Observé topo
Références locales
Entier nbse, nbst
Serie se
Station st1, st2

DEBUT
        nbse ← getNbSeries()
        se ← selectSerie(topo, nbse)
        nbst ← getNbStations(se)
        st1 ← selectStation(se, nbst)
        st2 ← selectStation(se, nbst - 1)
        Si getNumero(st1) >= getNumero(st2) Alors
        DEBUT
                Retourner getNumero(st1)
        FIN
        Sinon
        DEBUT
                Retourner getNumero(st2)
        FIN
    FIN

//Fonction retournant le nombre de visées qu'il y a au total dans la topographie
//Le nombre de visées est égal au nombre de stations, qui s'incrémente à chaque fois qu'une série commence et termine par une station connue

Observateur Topographie : getTotalVisees() : Entier
Observé topo
Références locales
Entier nbst, nbse, i
Serie se

DEBUT
    nbst ← getTotalStations(topo)
    nbse ← getNbSeries(topo)
    i ← 1
    TANTQUE i <= nbse FAIRE
    DEBUT
        se ← selectSerie(topo,i)
        Si (departExistant(se) et arriveeExistante(se)) Alors
        DEBUT
```

```
                    nbst ← nbst + 1
                FIN
                i ← i+1
        FIN
                Retourner nbst
FIN
```

Constructeur Matrice : R(topo) : Matrice

```
Références locales
Matrice r
Serie se
Entier i, j, nbse, nbst, m
Station st

DEBUT
        r ← creerMatrice(getTotalVisees(topo),getTotalStations(topo))
        nbse ← getNbSeries(topo)
        m ← 1
        i ← 1
        TANTQUE i <= nbse FAIRE
        DEBUT
                se ← getNbSeries(topo, i)
                nbst ← getNbStations(se)
                TANTQUE j <= nbst-1 FAIRE
                DEBUT
                        st1 ← selectStation(j)
                        st2 ← selectStation(j+1)
                   //Pour éviter de traiter le point d'entrée
                        Si  i!=1 et j!=1 Alors
                        DEBUT
                                setValeurs(r, m,getNumero(st1),-1)
                        FIN
                        setValeurs(r, m,getNumero(st2),1)
                        m ← m + 1
                        j ← j+1
                FIN
                i ← i+1
        FIN
        Retourner r
FIN
```

Constructeur Matrice : deltaX(topo) : Matrice

```
Références locales
        Matrice dX
        Station st
        Serie se
        Entier i, j, m
        Reel x

DEBUT
        dX ← creerMatrice(getTotalVisees(topo),1)
        m ← 1
        i ← 1
        TANTQUE i <= getNbSeries(topo) FAIRE
                DEBUT
                se ← getNbSeries(topo, i)
                j ← 2
                TANTQUE j <= getNbStations(se) FAIRE
                        DEBUT
                        st ← selectStation(j)
                        x ← getLongueur(st) * cos(recAngle(st),recUniteAngle(topo))
                        dX ← setValeurs(dX, m, 1, x)
                        m ← m + 1
                        j ← j+1
                        FIN
                i ← i+1
                FIN
        Retourner dX
FIN
```

Constructeur Matrice : WX(topo) : Matrice

```
Références locales
        Matrice w
        Entier i, j, m
        Reel e, r

DEBUT
        w ← creerMatrice(getTotalVisees(topo), getTotalVisees(topo))
        m ← 1
        i ← 1
        TANTQUE i <= getNbSeries(topo) FAIRE
                DEBUT
                se ← getNbSeries(topo, i)
                j ← 2
                TANTQUE j <= getNbStations(se) FAIRE
                        DEBUT
                        st ← selectStation(j)
```

```
                                e ←
abs(recPresicionLongueur(topo)*getLongueur(st)*cos(recAngle(st),recUniteAngle(topo)))
                                        +
abs(2*pi*recPresicionAngle(topo)*getLongueur(st)*sin(recAngle(st),recUniteAngle(topo)))
                                r ← 1 / (e * e)
                                w ← setValeurs(w, m, m, r)
                                m ← m + 1
                                j ← j+1
                                FIN
                        I ← i+1
                        FIN
        Retourner w
FIN
```

Constructeur Matrice : BX(topo) : Matrice

Références locales

```
DEBUT
        Retourner
multiplicationMatrice(multiplicationMatrice(transposee(R(topo)),WX(topo)),R(topo))
FIN
```

Constructeur Matrice : TX(topo) : Matrice

Références locales

```
DEBUT
        Retourner
multiplicationMatrice(multiplicationMatrice(transposee(R(topo)),WX(topo)),deltaX(topo))
FIN
```

Constructeur Matrice : VX(topo) : Matrice

Références locales

```
        Matrice m
        Entier i, j, k
        Reel r, v

DEBUT
        m ←creerMatrice(getTotalStations(topo),getTotalStations(topo))
        i ← 1
```

```
        TANTQUE i <= getTotalStations(topo) FAIRE
                DEBUT
                v ← 0
                Si i != 1 Alors
                        DEBUT
                        K ← 1
                        TANTQUE k <= i - 1 FAIRE
                                DEBUT
                                v ← v + getValeur(m, i, k)*getValeur(m, i, k)
                                k ← k+1
                                FIN
                        FIN
                r ← racine(getValeur(BX(topo), i, i) - v)
                m ← setValeurs(m, i, i, r)
                j ← i+1
                TANTQUE j <= getTotalStations(topo) FAIRE
                        DEBUT
                        v ← 0
                        Si i != 1 Alors
                                DEBUT
                                Pour k de 1 a i - 1 pas 1
                                        DEBUT
                                        v ← v + getValeur(m, i, k)*getValeur(m, j, k)
                                        FIN
                                FIN
                        r ← (getValeur(BX(topo), i, j) - v)/getValeur(m, i, i)
                        m ← setValeurs(m, j, i, r)
                        j ← j+1
            FIN
                I ← i+1
        FIN
        Retourner m
FIN
```

Constructeur Matrice : SX(topo) : Matrice

```
Références locales
        Matrice m
        Entier j, k
        Reel s, v


DEBUT
        m ← creerMatrice(1,getTotalStations(topo))
        s ← getValeur(TX(topo),1,1)/getValeur(VX(topo),1,1)
        m ← setValeurs(m, 1, 1, s)
        j ← 2
```

```
        TANTQUE j <= getTotalStations(topo) FAIRE
                DEBUT
                v ← 0
                j ← 1
                TANTQUE k <= j - 1 FAIRE
                        DEBUT
                        v ← v + getValeur(VX(topo), j, k)*getValeur(m, k, 1)
                        k ← k+1
                        FIN
                        s ← (getValeur(TX(topo),j,1) - v)/getValeur(VX(topo),j,j)
                        m ← setValeurs(m, j, 1, s)
                        j ← j+1
                FIN
        Retourner m
FIN
```

Constructeur Matrice : X(topo) : Matrice

```
Références locales
        Matrice m
        Entier nbst, j, k
        Reel x, v

DEBUT
        nbst ← getTotalStations(topo)
        m ← creerMatrice(nbst,1)
        x ← getValeur(SX(topo),nbst,1)/getValeur(VX(topo),nbst,nbst)
        m ← setValeurs(m, nbst, 1, x)
        j ← nbst - 1
        TantQue j > 0 Faire
                DEBUT
                v ← 0
                k ← j+1
                TANTQUE k <= nbst FAIRE
                        DEBUT
                        v ← v + getValeur(VX(topo), k, j)*getValeur(m, k, 1)
                        k ← k+1
                        FIN
                x ← (getValeur(SX(topo), j, 1) - v) / getValeur(VX(topo), j, j)
                m ← setValeurs(m, j, 1, x)
                j ← j - 1
                FIN
        Retourner m
FIN
```

Question 8 :

Observateur Topographie : sauvegarderPoints(Chaine nomFichier) :
Observé topo
Références locales :
Booleen reussi
Flot flot
Liste listeserie
Liste listestation
Serie seriecourante
Station stationcourante
Entier numeroliste
Entier numerostation

DEBUT

Flot ← creerFlot(Element e)
Ouvrir(flot,concat(nomFichier, ‘’.top’’))

Listeserie ← getListeSeries(topo)

TANTQUE NON estVide(listeserie) FAIRE
DEBUT
seriecourante ← premier(listeSerie)
Listestation ← getListeStation(seriecourante)
listeSerie ← reste(listeSerie)

TANTQUE NON estVide(listestation) FAIRE
DEBUT
Stationcourante ← premier(listestation)
Listestation ← reste(listestation)
Mettre(flot, getNumero(seriecourante))
Mettre(flot, ‘ ‘)
Mettre(flot,getNumero(stationcourante))
Mettre(flot, ‘ ‘)
Mettre(flot,getX(stationcourante))
Mettre(flot, ‘ ‘)
Mettre(flot,getY(stationcourante))
FIN

FIN
Fermer(flot)
FIN