Voyageur du commerce

Généré par Doxygen 1.8.6

Mardi 15 Avril 2014 11 :01 :14

Table des matières

Chapitre 1

Page principale

Cette application a pour vocation de calculer des solutions au problème du voyageur de commerce

Chapitre 2

Index des structures de données

2.1 Structures de données

Liste des structures de données avec une brève description :

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

Chapitre 4

Documentation des structures de données

4.1 Référence de la structure Matrice

Champs de données

- int **dimension**
- **point** ∗ **ref**
- float ∗∗ **tab**

4.1.1 Description détaillée

Définition à la ligne 11 du fichier matrice.c.

4.1.2 Documentation des champs

4.1.2.1 int dimension

Définition à la ligne 13 du fichier matrice.c.

4.1.2.2 point∗ ref

Définition à la ligne 14 du fichier matrice.c.

4.1.2.3 float∗∗ tab

Définition à la ligne 15 du fichier matrice.c.

La documentation de cette structure a été générée à partir du fichier suivant :

- **matrice.c**

4.2 Référence de la structure Point

Champs de données

- signed int **x**
- signed int **y**
- bool **visited**

4.2.1 Description détaillée

Définition à la ligne 10 du fichier point.c.

4.2.2 Documentation des champs

4.2.2.1 bool visited

Définition à la ligne 14 du fichier point.c.

4.2.2.2 signed int x

Définition à la ligne 12 du fichier point.c.

4.2.2.3 signed int y

Définition à la ligne 13 du fichier point.c.

La documentation de cette structure a été générée à partir du fichier suivant :
- **point.c**

Chapitre 5

Documentation des fichiers

5.1 Référence du fichier main.c

Point (p. 7) d'entrée de l'application.

```
#include <stdio.h>
#include <stdlib.h>
#include "point.h"
#include "projetAlgo.h"
#include "matrice.h"
#include "tspIOtourO.h"
```

Fonctions

- void **erreurArguments** ()
- int **main** (int argc, char ∗∗argv)
  Fonction principale permettant à l'utilisateur de choisir une solution de parcours depuis un fichier tsp.

5.1.1 Description détaillée

Point (p. 7) d'entrée de l'application.

Définition dans le fichier **main.c**.

5.1.2 Documentation des fonctions

5.1.2.1 void erreurArguments ( )

Définition à la ligne 137 du fichier main.c.

5.1.2.2 int main (int *argc,* char ∗∗ *argv*)

Fonction principale permettant à l'utilisateur de choisir une solution de parcours depuis un fichier tsp.

Paramètres

argc	contient le nombre d'argument passé en parametre
argv	contient les arguments de l'appel sous forme de chaine de caractere

Renvoie

: EXIT_FAILURE si il y a un probleme, EXIT_SUCESS si tout ce passe bien

Définition à la ligne 27 du fichier main.c.

5.2 Référence du fichier matrice.c

```
#include "point.h"
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <string.h>
#include "matrice.h"
```

Structures de données

— struct **Matrice**

Fonctions

— **matrice creerMatriceVide** (int dimension)
— **matrice creerMatriceDesPoints** (**point** liste[], int dimension)
— void **detruireMatrice** (**matrice** m)
— **matrice cloneMatrice** (**matrice** m)
— int **getIndicePoint** (**matrice** m, **point** p)
— **point getPointIndice** (**matrice** m, int indice)
— float **getDistanceIndice** (**matrice** m, int ref1, int ref2)
— float **getDistancePoint** (**matrice** m, **point** p1, **point** p2)
— int **getDimensionMatrice** (**matrice** m)
— void **setDistanceIndice** (**matrice** m, int ref1, int ref2, float distance)
— void **setPointIndice** (**matrice** m, int i, **point** p)
— **point** ∗ **getTableauPointsMatrice** (**matrice** m)
— void **afficherMatrice** (**matrice** m)
— int **findMin** (float ∗list, int len, int k)
— int **lowerBound** (**matrice** m)
— void **markAsInfinite** (**matrice** m, int i, int j)

5.2.1 Documentation des fonctions

5.2.1.1 void afficherMatrice (matrice *m*)

Définition à la ligne 125 du fichier matrice.c.

5.2.1.2 matrice cloneMatrice (matrice *m*)

Définition à la ligne 65 du fichier matrice.c.

5.2.1.3 matrice creerMatriceDesPoints (point *liste[],* int *dimension*)

Définition à la ligne 39 du fichier matrice.c.

5.2.1.4 matrice creerMatriceVide (int *dimension*)

Définition à la ligne 23 du fichier matrice.c.

5.2.1.5 void detruireMatrice (matrice *m*)

Définition à la ligne 56 du fichier matrice.c.

5.2.1.6 int findMin (float * *list,* int *len,* int *k*)

Définition à la ligne 146 du fichier matrice.c.

5.2.1.7 int getDimensionMatrice (matrice *m*)

Définition à la ligne 100 du fichier matrice.c.

5.2.1.8 float getDistanceIndice (matrice *m,* int *ref1,* int *ref2*)

Définition à la ligne 88 du fichier matrice.c.

5.2.1.9 float getDistancePoint (matrice *m,* point *p1,* point *p2*)

Définition à la ligne 94 du fichier matrice.c.

5.2.1.10 int getIndicePoint (matrice *m,* point *p*)

Définition à la ligne 73 du fichier matrice.c.

5.2.1.11 point getPointIndice (matrice *m,* int *indice*)

Définition à la ligne 82 du fichier matrice.c.

5.2.1.12 point* getTableauPointsMatrice (matrice *m*)

Définition à la ligne 117 du fichier matrice.c.

5.2.1.13 int lowerBound (matrice *m*)

Définition à la ligne 160 du fichier matrice.c.

5.2.1.14 void markAsInfinite (matrice *m,* int *i,* int *j*)

Définition à la ligne 192 du fichier matrice.c.

5.2.1.15 void setDistanceIndice (matrice *m,* int *ref1,* int *ref2,* float *distance*)

Définition à la ligne 105 du fichier matrice.c.

5.2.1.16 void setPointIndice (**matrice** *m,* **int** *i,* **point** *p*)

Définition à la ligne 112 du fichier matrice.c.

5.3 Référence du fichier matrice.h

```
#include "point.h"
#include <stdio.h>
```

Définitions de type

— typedef struct **Matrice** ∗ **matrice**

Fonctions

— **matrice creerMatriceDesPoints** (**point** liste[], int dimension)
— **matrice creerMatriceVide** (int dimension)
— void **detruireMatrice** (**matrice** m)
— **matrice cloneMatrice** (**matrice** m)
— int **getIndicePoint** (**matrice** m, **point** p)
— **point getPointIndice** (**matrice** m, int indice)
— float **getDistanceIndice** (**matrice** m, int ref1, int ref2)
— float **getDistancePoint** (**matrice** m, **point** p1, **point** p2)
— **point** ∗ **getTableauPointsMatrice** (**matrice** m)
— void **setDistanceIndice** (**matrice** m, int ref1, int ref2, float distance)
— void **setPointIndice** (**matrice** m, int i, **point** p)
— int **getDimensionMatrice** (**matrice** m)
— void **afficherMatrice** (**matrice** m)
— **matrice creerMatriceTSP** (char ∗fnom)
— int **lowerBound** (**matrice** m)
— void **markAsInfinite** (**matrice** m, int i, int j)

5.3.1 Documentation des définitions de type

5.3.1.1 typedef struct Matrice∗ matrice

Définition à la ligne 7 du fichier matrice.h.

5.3.2 Documentation des fonctions

5.3.2.1 void afficherMatrice (**matrice** *m*)

Définition à la ligne 125 du fichier matrice.c.

5.3.2.2 matrice cloneMatrice (**matrice** *m*)

Définition à la ligne 65 du fichier matrice.c.

5.3.2.3 matrice creerMatriceDesPoints (**point** *liste[],* **int** *dimension*)

Définition à la ligne 39 du fichier matrice.c.

5.3.2.4 matrice creerMatriceTSP (char * *fnom*)

Définition à la ligne 11 du fichier tspIOtourO.c.

5.3.2.5 matrice creerMatriceVide (int *dimension*)

Définition à la ligne 23 du fichier matrice.c.

5.3.2.6 void detruireMatrice (matrice *m*)

Définition à la ligne 56 du fichier matrice.c.

5.3.2.7 int getDimensionMatrice (matrice *m*)

Définition à la ligne 100 du fichier matrice.c.

5.3.2.8 float getDistanceIndice (matrice *m,* int *ref1,* int *ref2*)

Définition à la ligne 88 du fichier matrice.c.

5.3.2.9 float getDistancePoint (matrice *m,* point *p1,* point *p2*)

Définition à la ligne 94 du fichier matrice.c.

5.3.2.10 int getIndicePoint (matrice *m,* point *p*)

Définition à la ligne 73 du fichier matrice.c.

5.3.2.11 point getPointIndice (matrice *m,* int *indice*)

Définition à la ligne 82 du fichier matrice.c.

5.3.2.12 point* getTableauPointsMatrice (matrice *m*)

Définition à la ligne 117 du fichier matrice.c.

5.3.2.13 int lowerBound (matrice *m*)

Définition à la ligne 160 du fichier matrice.c.

5.3.2.14 void markAsInfinite (matrice *m,* int *i,* int *j*)

Définition à la ligne 192 du fichier matrice.c.

5.3.2.15 void setDistanceIndice (matrice *m,* int *ref1,* int *ref2,* float *distance*)

Définition à la ligne 105 du fichier matrice.c.

5.3.2.16 void setPointIndice (matrice *m,* int *i,* point *p*)

Définition à la ligne 112 du fichier matrice.c.

5.4 Référence du fichier point.c

```
#include <stdlib.h>
#include <assert.h>
#include <math.h>
#include <stdbool.h>
#include <stdio.h>
#include "point.h"
```

Structures de données

— struct **Point**

Fonctions

— **point creerPoint** (signed int x, signed int y)
— void **detruirePoint** (**point** p)
— bool **equals** (**point** p1, **point** p2)
— **point clone** (**point** p)
— signed int **getX** (**point** p)
— signed int **getY** (**point** p)
— bool **isVisited** (**point** p)
— void **setX** (**point** p, signed int x)
— void **setY** (**point** p, signed int y)
— void **markVisited** (**point** p)
— void **markNoVisited** (**point** p)
— void **afficherPoint** (**point** p)
— void **afficherListeDesPoints** (**point** ∗p, int len)
— float **distanceManhattan** (**point** p1, **point** p2)
— float **distanceEntreDeuxPoints** (**point** p1, **point** p2)

5.4.1 Documentation des fonctions

5.4.1.1 void afficherListeDesPoints (point ∗ *p,* int *len*)

Définition à la ligne 97 du fichier point.c.

5.4.1.2 void afficherPoint (point *p*)

Définition à la ligne 90 du fichier point.c.

5.4.1.3 point clone (point *p*)

Définition à la ligne 45 du fichier point.c.

5.4.1.4 point creerPoint (signed int *x,* signed int *y*)

Définition à la ligne 22 du fichier point.c.

5.4.1.5 void detruirePoint (point *p*)

Définition à la ligne 33 du fichier point.c.

5.4.1.6 float distanceEntreDeuxPoints (point *p1*, point *p2*)

Définition à la ligne 110 du fichier point.c.

5.4.1.7 float distanceManhattan (point *p1*, point *p2*)

Définition à la ligne 104 du fichier point.c.

5.4.1.8 bool equals (point *p1*, point *p2*)

Définition à la ligne 40 du fichier point.c.

5.4.1.9 signed int getX (point *p*)

Définition à la ligne 52 du fichier point.c.

5.4.1.10 signed int getY (point *p*)

Définition à la ligne 57 du fichier point.c.

5.4.1.11 bool isVisited (point *p*)

Définition à la ligne 62 du fichier point.c.

5.4.1.12 void markNoVisited (point *p*)

Définition à la ligne 85 du fichier point.c.

5.4.1.13 void markVisited (point *p*)

Définition à la ligne 79 du fichier point.c.

5.4.1.14 void setX (point *p*, signed int *x*)

Définition à la ligne 67 du fichier point.c.

5.4.1.15 void setY (point *p*, signed int *y*)

Définition à la ligne 73 du fichier point.c.

5.5 Référence du fichier point.h

```
#include <stdbool.h>
```

Définitions de type

— typedef struct **Point** ∗ **point**

Fonctions

— **point creerPoint** (signed int x, signed int y)
— void **detruirePoint** (**point** p)
— bool **equals** (**point** p1, **point** p2)
— **point clone** (**point** p)
— signed int **getX** (**point** p)
— signed int **getY** (**point** p)
— bool **isVisited** (**point** p)
— void **setX** (**point** p, signed int x)
— void **setY** (**point** p, signed int y)
— void **markVisited** (**point** p)
— void **markNoVisited** (**point** p)
— void **afficherPoint** (**point** p)
— void **afficherListeDesPoints** (**point** ∗p, int len)
— float **distanceEntreDeuxPoints** (**point** p1, **point** p2)
— float **distanceManhattan** (**point** p1, **point** p2)

5.5.1 Documentation des définitions de type

5.5.1.1 typedef struct Point∗ point

Définition à la ligne 6 du fichier point.h.

5.5.2 Documentation des fonctions

5.5.2.1 void afficherListeDesPoints (point ∗ *p,* int *len*)

Définition à la ligne 97 du fichier point.c.

5.5.2.2 void afficherPoint (point *p*)

Définition à la ligne 90 du fichier point.c.

5.5.2.3 point clone (point *p*)

Définition à la ligne 45 du fichier point.c.

5.5.2.4 point creerPoint (signed int *x,* signed int *y*)

Définition à la ligne 22 du fichier point.c.

5.5.2.5 void detruirePoint (point *p*)

Définition à la ligne 33 du fichier point.c.

5.5.2.6 float distanceEntreDeuxPoints (point *p1,* point *p2*)

Définition à la ligne 110 du fichier point.c.

5.5.2.7 float distanceManhattan (point *p1*, point *p2*)

Définition à la ligne 104 du fichier point.c.

5.5.2.8 bool equals (point *p1*, point *p2*)

Définition à la ligne 40 du fichier point.c.

5.5.2.9 signed int getX (point *p*)

Définition à la ligne 52 du fichier point.c.

5.5.2.10 signed int getY (point *p*)

Définition à la ligne 57 du fichier point.c.

5.5.2.11 bool isVisited (point *p*)

Définition à la ligne 62 du fichier point.c.

5.5.2.12 void markNoVisited (point *p*)

Définition à la ligne 85 du fichier point.c.

5.5.2.13 void markVisited (point *p*)

Définition à la ligne 79 du fichier point.c.

5.5.2.14 void setX (point *p*, signed int *x*)

Définition à la ligne 67 du fichier point.c.

5.5.2.15 void setY (point *p*, signed int *y*)

Définition à la ligne 73 du fichier point.c.

5.6 Référence du fichier projetAlgo.c

```
#include "point.h"
#include <stdlib.h>
#include "matrice.h"
```

Fonctions

- int **PointLePlusProche** (int indicePointActuel, **matrice** m)
  *Cherche dans la matrice quel est le point non visité le plus proche du point indiqué par l'indice, cette fonction utilise la fonction **getDistanceIndice()** (p. 11) permettant de connaitre la distance entre deux points.*
- **point** ∗ **nearestNeighbour** (**matrice** mIn)

Parcour de la matrice, crée un ordre de parcours permettant d'effectuer une distance moindre, recherche de point le plus proche.

- int **overallDistance** (**matrice** m, **point** ∗points)
- int **overallDistanceVerbose** (**matrice** m, **point** ∗points)
- void **swap** (**point** ∗plist, int i, int j)
- void **copyList** (**point** ∗pIn, **point** ∗pOut, int len)
- void **copyListIndice** (**point** ∗pIn, **point** ∗pOut, int start, int end)
- void **deleteFromList** (**point** ∗in, int length, **point** p)
- void **bruteForceRough** (**matrice** m, **point** ∗pIn, int i, int n, int ∗min, **point** ∗pOut)
- **point** ∗ **bruteForce** (**matrice** m)
- int **prim** (**matrice** mIn, **point** ∗TabVisite)
- **point** ∗ **branchBound** (**matrice** m)

5.6.1 Documentation des fonctions

5.6.1.1 point∗ branchBound (matrice *m*)

Définition à la ligne 282 du fichier projetAlgo.c.

5.6.1.2 point∗ bruteForce (matrice *m*)

Définition à la ligne 209 du fichier projetAlgo.c.

5.6.1.3 void bruteForceRough (matrice *m,* point ∗ *pIn,* int *i,* int *n,* int ∗ *min,* point ∗ *pOut*)

Définition à la ligne 182 du fichier projetAlgo.c.

5.6.1.4 void copyList (point ∗ *pIn,* point ∗ *pOut,* int *len*)

Définition à la ligne 150 du fichier projetAlgo.c.

5.6.1.5 void copyListIndice (point ∗ *pIn,* point ∗ *pOut,* int *start,* int *end*)

Définition à la ligne 157 du fichier projetAlgo.c.

5.6.1.6 void deleteFromList (point ∗ *in,* int *length,* point *p*)

Définition à la ligne 166 du fichier projetAlgo.c.

5.6.1.7 point∗ nearestNeighbour (matrice *mIn*)

Parcour de la matrice, crée un ordre de parcours permettant d'effectuer une distance moindre, recherche de point le plus proche.

Paramètres

m	matrice d'entrée qui permet d'obtenir la liste des points que l'on souhaite parcourir.

Renvoie

ordreDePassage[] tableau de sortie qui contiendra le nouveau parcours

Définition à la ligne 67 du fichier projetAlgo.c.

5.6.1.8 int overallDistance (**matrice** *m,* **point** ∗ *points*)

Définition à la ligne 104 du fichier projetAlgo.c.

5.6.1.9 int overallDistanceVerbose (**matrice** *m,* **point** ∗ *points*)

Définition à la ligne 116 du fichier projetAlgo.c.

5.6.1.10 int PointLePlusProche (int *indicePointActuel,* **matrice** *m*)

Cherche dans la matrice quel est le point non visité le plus proche du point indiqué par l'indice, cette fonction utilise la fonction **getDistanceIndice()** (p. 11) permettant de connaitre la distance entre deux points.

Paramètres

indicePoint-Actuel	correspond à l'indice du point actuel, permettant l'accès à ce point.
m	matrice contenant tous les points

Renvoie

: indice du point le plus proche du point actuel

Définition à la ligne 14 du fichier projetAlgo.c.

5.6.1.11 int prim (**matrice** *mIn,* **point** ∗ *TabVisite*)

Définition à la ligne 231 du fichier projetAlgo.c.

5.6.1.12 void swap (**point** ∗ *plist,* int *i,* int *j*)

Définition à la ligne 138 du fichier projetAlgo.c.

5.7 Référence du fichier projetAlgo.h

```
#include "matrice.h"
```

Fonctions

- int **PointLePlusProche** (int indicePointActuel, **matrice** m)
  *Cherche dans la matrice quel est le point non visité le plus proche du point indiqué par l'indice, cette fonction utilise la fonction **getDistanceIndice()** (p. 11) permettant de connaitre la distance entre deux points.*
- **point** ∗ **nearestNeighbour** (**matrice** m)
  Parcour de la matrice, crée un ordre de parcours permettant d'effectuer une distance moindre, recherche de point le plus proche.
- int **overallDistance** (**matrice** m, **point** ∗points)
- int **overallDistanceVerbose** (**matrice** m, **point** ∗points)
- **point** ∗ **bruteForce** (**matrice** m)
- void **copyList** (**point** ∗pIn, **point** ∗pOut, int len)
- void **copyListIndice** (**point** ∗pIn, **point** ∗pOut, int start, int end)
- **point** ∗ **branchBound** (**matrice** m)
- int **prim** (**matrice** m, **point** ∗TabVisite)

5.7.1 Documentation des fonctions

5.7.1.1 point* branchBound (matrice *m*)

Définition à la ligne 282 du fichier projetAlgo.c.

5.7.1.2 point* bruteForce (matrice *m*)

Définition à la ligne 209 du fichier projetAlgo.c.

5.7.1.3 void copyList (point * *pIn*, point * *pOut*, int *len*)

Définition à la ligne 150 du fichier projetAlgo.c.

5.7.1.4 void copyListIndice (point * *pIn*, point * *pOut*, int *start*, int *end*)

Définition à la ligne 157 du fichier projetAlgo.c.

5.7.1.5 point* nearestNeighbour (matrice *mIn*)

Parcour de la matrice, crée un ordre de parcours permettant d'effectuer une distance moindre, recherche de point le plus proche.

Paramètres

m	matrice d'entrée qui permet d'obtenir la liste des points que l'on souhaite parcourir.

Renvoie

ordreDePassage[] tableau de sortie qui contiendra le nouveau parcours

Définition à la ligne 67 du fichier projetAlgo.c.

5.7.1.6 int overallDistance (matrice *m*, point * *points*)

Définition à la ligne 104 du fichier projetAlgo.c.

5.7.1.7 int overallDistanceVerbose (matrice *m*, point * *points*)

Définition à la ligne 116 du fichier projetAlgo.c.

5.7.1.8 int PointLePlusProche (int *indicePointActuel*, matrice *m*)

Cherche dans la matrice quel est le point non visité le plus proche du point indiqué par l'indice, cette fonction utilise la fonction **getDistanceIndice()** (p. 11) permettant de connaitre la distance entre deux points.

Paramètres

indicePoint-Actuel	correspond à l'indice du point actuel, permettant l'accès à ce point.

m	matrice contenant tous les points

Renvoie

: indice du point le plus proche du point actuel

Définition à la ligne 14 du fichier projetAlgo.c.

5.7.1.9 int prim (**matrice** *m,* **point** ∗ *TabVisite*)

Définition à la ligne 231 du fichier projetAlgo.c.

5.8 Référence du fichier test_BF.c

```
#include <stdio.h>
#include <stdlib.h>
#include "point.h"
#include "projetAlgo.h"
#include "matrice.h"
#include "tspIOtour0.h"
```

Macros

— #define **nombreDePoints** 4

Fonctions

— int **main** ()

5.8.1 Documentation des macros

5.8.1.1 #define nombreDePoints 4

Définition à la ligne 8 du fichier test_BF.c.

5.8.2 Documentation des fonctions

5.8.2.1 int main ( )

Définition à la ligne 9 du fichier test_BF.c.

5.9 Référence du fichier test_NN.c

```
#include <stdio.h>
#include <stdlib.h>
#include "point.h"
#include "projetAlgo.h"
#include "matrice.h"
#include "tspIOtour0.h"
```

Fonctions

— int **main** ()

5.9.1 Documentation des fonctions

5.9.1.1 int main ( )

Définition à la ligne 8 du fichier test_NN.c.

5.10 Référence du fichier test_Prim.c

```
#include <stdio.h>
#include <stdlib.h>
#include "point.h"
#include "projetAlgo.h"
#include "matrice.h"
#include "tspIOtourO.h"
```

Macros

— #define **nombreDePoints** 4

Fonctions

— int **main** ()

5.10.1 Documentation des macros

5.10.1.1 #define nombreDePoints 4

Définition à la ligne 8 du fichier test_Prim.c.

5.10.2 Documentation des fonctions

5.10.2.1 int main ( )

Définition à la ligne 9 du fichier test_Prim.c.

5.11 Référence du fichier tspIOtourO.c

```
#include <stdio.h>
#include <assert.h>
#include <stdlib.h>
#include <string.h>
#include "point.h"
#include "matrice.h"
#include "tspIOtourO.h"
```

Macros

— #define **_GNU_SOURCE**

Fonctions

— **matrice creerMatriceTSP** (char ∗fileName)
— void **creerTSPMatrice** (char ∗fileName, **matrice** m)
— void **creerTOUR** (char ∗fileName, **matrice** m, **point** liste[])

5.11.1 Documentation des macros

5.11.1.1 #define _GNU_SOURCE

Définition à la ligne 1 du fichier tspIOtourO.c.

5.11.2 Documentation des fonctions

5.11.2.1 matrice creerMatriceTSP (char ∗ *fileName*)

Définition à la ligne 11 du fichier tspIOtourO.c.

5.11.2.2 void creerTOUR (char ∗ *fileName,* matrice *m,* point *liste[]*)

Définition à la ligne 90 du fichier tspIOtourO.c.

5.11.2.3 void creerTSPMatrice (char ∗ *fileName,* matrice *m*)

Définition à la ligne 61 du fichier tspIOtourO.c.

5.12 Référence du fichier tspIOtourO.h

```
#include <stdio.h>
#include "matrice.h"
#include "point.h"
```

Fonctions

— **matrice creerMatriceTSP** (char ∗fnom)
— void **creerTSPMatrice** (char ∗fnom, **matrice** m)
— void **creerTOUR** (char ∗fnom, **matrice** m, **point** liste[])

5.12.1 Documentation des fonctions

5.12.1.1 matrice creerMatriceTSP (char ∗ *fnom*)

Définition à la ligne 11 du fichier tspIOtourO.c.

5.12.1.2 void creerTOUR (char ∗ *fnom,* matrice *m,* point *liste[]*)

Définition à la ligne 90 du fichier tspIOtourO.c.

5.12.1.3 void creerTSPMatrice (char * *fnom*, matrice *m*)

Définition à la ligne 61 du fichier tspIOtourO.c.

Index