

# Le voyageur de commerce

1

Généré par Doxygen 1.7.1

Thu Jan 10 2013 11 :42 :06



# Table des matières

<b>1</b>	<b>Le problème du voyageur de commerce - Projet d'algorithmique en langage C</b>	<b>1</b>
1.1	problème . . . . .	1
1.2	algorithmes implémenté . . . . .	1
<b>2</b>	<b>Index des structures de données</b>	<b>3</b>
2.1	Structures de données . . . . .	3
<b>3</b>	<b>Index des fichiers</b>	<b>5</b>
3.1	Liste des fichiers . . . . .	5
<b>4</b>	<b>Documentation des structures de données</b>	<b>7</b>
4.1	Référence de la structure Algo . . . . .	7
4.1.1	Description détaillée . . . . .	7
4.2	Référence de la structure Distance . . . . .	7
4.2.1	Description détaillée . . . . .	8
4.3	Référence de la structure Errors . . . . .	8
4.3.1	Description détaillée . . . . .	9
4.4	Référence de la structure Instance . . . . .	9
4.4.1	Description détaillée . . . . .	9
4.5	Référence de la structure Tour . . . . .	10
4.5.1	Description détaillée . . . . .	10
4.6	Référence de la structure Town . . . . .	10
4.6.1	Description détaillée . . . . .	11
<b>5</b>	<b>Documentation des fichiers</b>	<b>13</b>
5.1	Référence du fichier lib/bruteForce.h . . . . .	13
5.1.1	Description détaillée . . . . .	13
5.1.2	Documentation des fonctions . . . . .	13
5.1.2.1	bruteForce_bestPath . . . . .	13
5.2	Référence du fichier lib/distance.h . . . . .	14

5.2.1	Description détaillée . . . . .	14
5.2.2	Documentation des fonctions . . . . .	14
5.2.2.1	distance_betweenTowns . . . . .	14
5.2.2.2	distance_calculDistance . . . . .	15
5.2.2.3	distance_new . . . . .	15
5.2.2.4	distance_searchDistance . . . . .	15
5.3	Référence du fichier lib/errors.h . . . . .	15
5.3.1	Description détaillée . . . . .	16
5.3.2	Documentation des fonctions . . . . .	17
5.3.2.1	errors_displayErrorsMessage . . . . .	17
5.3.2.2	errors_new . . . . .	17
5.3.2.3	errors_setFileNotFound . . . . .	17
5.3.2.4	errors_setMissingParameterGa . . . . .	17
5.3.2.5	errors_setMissingParameterLsnr . . . . .	17
5.3.2.6	errors_setMissingParameterLsr . . . . .	17
5.3.2.7	errors_setNbArguments . . . . .	18
5.3.2.8	errors_setNoAlgoSpecified . . . . .	18
5.3.2.9	errors_setNoValidParameterLsnr . . . . .	18
5.3.2.10	errors_setNoValidParameterLsr . . . . .	18
5.3.2.11	errors_setTagFNotFound . . . . .	18
5.4	Référence du fichier lib/instance.h . . . . .	18
5.4.1	Description détaillée . . . . .	19
5.4.2	Documentation des fonctions . . . . .	19
5.4.2.1	instance_display . . . . .	19
5.4.2.2	instance_displayLinearVector . . . . .	20
5.4.2.3	instance_displayMatrix . . . . .	20
5.4.2.4	instance_initializeDistancesMatrix . . . . .	20
5.4.2.5	instance_new . . . . .	20
5.4.2.6	instance_push . . . . .	20
5.5	Référence du fichier lib/localSearch.h . . . . .	21
5.5.1	Description détaillée . . . . .	21
5.6	Référence du fichier lib/parsing.h . . . . .	21
5.6.1	Description détaillée . . . . .	22
5.6.2	Documentation du type de l'énumération . . . . .	22
5.6.2.1	AlgoType . . . . .	22
5.6.3	Documentation des fonctions . . . . .	22

5.6.3.1	<code>parsing_algoType</code>	22
5.6.3.2	<code>parsing_parseFileName</code>	23
5.6.3.3	<code>parsing_parseVerboseMode</code>	23
5.7	Référence du fichier <code>lib/tour.h</code>	23
5.7.1	Description détaillée	24
5.7.2	Documentation des fonctions	24
5.7.2.1	<code>tour_2opt</code>	24
5.7.2.2	<code>tour_calculLength</code>	24
5.7.2.3	<code>tour_display</code>	25
5.7.2.4	<code>tour_new</code>	25
5.7.2.5	<code>tour_nextPermutation</code>	25
5.7.2.6	<code>tour_randomWalk</code>	25
5.8	Référence du fichier <code>lib/town.h</code>	25
5.8.1	Description détaillée	26
5.8.2	Documentation des fonctions	26
5.8.2.1	<code>town_new</code>	26
5.9	Référence du fichier <code>lib/util.h</code>	26
5.9.1	Description détaillée	27
5.9.2	Documentation des fonctions	27
5.9.2.1	<code>util_displayArray</code>	27
5.9.2.2	<code>util_rand</code>	27
5.9.2.3	<code>util_reverseArray</code>	28
5.9.2.4	<code>util_searchFirstOccurenceInArray</code>	28
5.9.2.5	<code>util_sousTabExist</code>	28
5.9.2.6	<code>util_sum</code>	29
5.9.2.7	<code>util_swap</code>	29
5.10	Référence du fichier <code>src/bruteForce.c</code>	29
5.10.1	Description détaillée	29
5.10.2	Documentation des fonctions	30
5.10.2.1	<code>bruteForce_bestPath</code>	30
5.11	Référence du fichier <code>src/distance.c</code>	30
5.11.1	Description détaillée	30
5.11.2	Documentation des fonctions	31
5.11.2.1	<code>distance_betweenTowns</code>	31
5.11.2.2	<code>distance_calculDistance</code>	31
5.11.2.3	<code>distance_new</code>	31

5.11.2.4	<code>distance_searchDistance</code>	31
5.12	Référence du fichier <code>src/genetic.c</code>	32
5.12.1	Description détaillée	32
5.13	Référence du fichier <code>src/instance.c</code>	32
5.13.1	Description détaillée	33
5.13.2	Documentation des fonctions	33
5.13.2.1	<code>instance_display</code>	33
5.13.2.2	<code>instance_displayLinearVector</code>	33
5.13.2.3	<code>instance_displayMatrix</code>	33
5.13.2.4	<code>instance_initializeDistancesMatrix</code>	33
5.13.2.5	<code>instance_new</code>	34
5.13.2.6	<code>instance_push</code>	34
5.14	Référence du fichier <code>src/localSearch.c</code>	34
5.14.1	Description détaillée	34
5.15	Référence du fichier <code>src/main.c</code>	35
5.15.1	Description détaillée	35
5.15.2	Documentation des fonctions	35
5.15.2.1	<code>main</code>	35
5.16	Référence du fichier <code>src/parsing.c</code>	35
5.16.1	Description détaillée	36
5.16.2	Documentation des fonctions	36
5.16.2.1	<code>parsing_algoType</code>	36
5.16.2.2	<code>parsing_parseFileName</code>	36
5.16.2.3	<code>parsing_parseVerboseMode</code>	37
5.17	Référence du fichier <code>src/tour.c</code>	37
5.17.1	Description détaillée	37
5.17.2	Documentation des fonctions	38
5.17.2.1	<code>tour_2opt</code>	38
5.17.2.2	<code>tour_calculLength</code>	38
5.17.2.3	<code>tour_display</code>	38
5.17.2.4	<code>tour_new</code>	38
5.17.2.5	<code>tour_nextPermutation</code>	38
5.17.2.6	<code>tour_randomWalk</code>	39
5.18	Référence du fichier <code>src/town.c</code>	39
5.18.1	Description détaillée	39
5.18.2	Documentation des fonctions	39

---

5.18.2.1	<a href="#">town_new</a>	39
5.19	<a href="#">Référence du fichier src/util.c</a>	39
5.19.1	<a href="#">Description détaillée</a>	40
5.19.2	<a href="#">Documentation des fonctions</a>	40
5.19.2.1	<a href="#">util_displayArray</a>	40
5.19.2.2	<a href="#">util_rand</a>	40
5.19.2.3	<a href="#">util_reverseArray</a>	41
5.19.2.4	<a href="#">util_searchFirstOccurenceInArray</a>	41
5.19.2.5	<a href="#">util_sousTabExist</a>	41
5.19.2.6	<a href="#">util_sum</a>	42
5.19.2.7	<a href="#">util_swap</a>	42





# Chapitre 1

## Le problème du voyageur de commerce - Projet d'algorithmique en langage C

### Auteur

L2 Antoine de Roquemaurel (G1.1)

### 1.1 problème

Étant donné  $n$  points (des "villes") et les distances les séparant, trouver un chemin de longueur totale qui passe exactement une fois par chaque point et reviennent au point de départ (une tournée).

Ce problème peut servir tel quel à l'optimisation de trajectoires de machines-outils : par exemple, pour minimiser le temps total que met une fraiseuse à commande numérique pour percer  $n$  points dans une plaque de tôle ou pour percer les trous des composants d'un circuit électronique comme dans le cas qui nous intéresse.

Ce problème est plus compliqué qu'il n'y paraît et on ne connaît pas de méthode de résolution permettant d'obtenir des solutions exactes en un temps raisonnable pour de grandes instances (grand nombre de villes) du problème. Pour ces grandes instances, on devra donc souvent se contenter de solutions approchées, car on se retrouve face à une explosion combinatoire : le nombre de chemins possibles passant par 69 villes est déjà un nombre d'une longueur de 100 chiffres. Pour comparaison, un nombre d'une longueur de 80 chiffres permettrait déjà de représenter le nombre d'atomes dans tout l'univers connu.

Le problème du "voyageur de commerce" a été étudié depuis longtemps et on dispose d'une grande variété d'algorithmes donnant le plus souvent des solutions approchées mais calculables en un temps raisonnable.

### 1.2 algorithmes implémentés

Ce problème sera implémenté via différents algorithmes :

- Brute force
- Recherche locale aléatoire
- Recherche locale systématique
- Algorithme génétique



# Chapitre 2

## Index des structures de données

### 2.1 Structures de données

Liste des structures de données avec une brève description :

<a href="#">Algo</a> (Enumération d'un algorithme ) . . . . .	7
<a href="#">Distance</a> (Objet des distances ) . . . . .	7
<a href="#">Errors</a> (Objet des erreurs ) . . . . .	8
<a href="#">Instance</a> (Objet des instances ) . . . . .	9
<a href="#">Tour</a> (Objet d'une tournée ) . . . . .	10
<a href="#">Town</a> (Objet des ville ) . . . . .	10



# Chapitre 3

## Index des fichiers

### 3.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

lib/ <a href="#">bruteForce.h</a> (Fonctions de brute force ) . . . . .	13
lib/ <a href="#">distance.h</a> (Fonctions des distances ) . . . . .	14
lib/ <a href="#">errors.h</a> (Fonctions sur les erreurs ) . . . . .	15
lib/ <a href="#">genetic.h</a> . . . . .	??
lib/ <a href="#">instance.h</a> (Fonctions sur les instances ) . . . . .	18
lib/ <a href="#">localSearch.h</a> (Fonctions utiles ) . . . . .	21
lib/ <a href="#">parsing.h</a> (Fonctions de parsing des arguments ) . . . . .	21
lib/ <a href="#">tour.h</a> (Fonctions des tournées ) . . . . .	23
lib/ <a href="#">town.h</a> (Fonctions des villes ) . . . . .	25
lib/ <a href="#">util.h</a> (Fonctions utiles ) . . . . .	26
src/ <a href="#">bruteForce.c</a> (Fonctions de brute force ) . . . . .	29
src/ <a href="#">distance.c</a> (Fonctions des distances ) . . . . .	30
src/ <a href="#">genetic.c</a> (Fonctions d'algorithmes génétiques ) . . . . .	32
src/ <a href="#">instance.c</a> (Fonctions utiles ) . . . . .	32
src/ <a href="#">localSearch.c</a> (Fonctions de recherche locale ) . . . . .	34
src/ <a href="#">main.c</a> (Fonction main ) . . . . .	35
src/ <a href="#">parsing.c</a> (Fonctions de parsing des arguments ) . . . . .	35
src/ <a href="#">tour.c</a> (Fonctions des tournées ) . . . . .	37
src/ <a href="#">town.c</a> (Fonctions des villes ) . . . . .	39
src/ <a href="#">util.c</a> (Fonctions utiles ) . . . . .	39



## Chapitre 4

# Documentation des structures de données

### 4.1 Référence de la structure Algo

Enumération d'un algorithme.

```
#include <parsing.h>
```

#### Champs de données

- [AlgoType](#) type  
*Le type de l'algorithme.*
- int [firstParameter](#)  
*Premier paramètre de l'algorithme.*
- int [secondParameter](#)  
*Second paramètre.*

#### 4.1.1 Description détaillée

Enumération d'un algorithme. Énumération d'un algorithme, contient le type de l'algorithme avec les eventuels paramètres

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

- [lib/parsing.h](#)

### 4.2 Référence de la structure Distance

Objet des distances.

```
#include <distance.h>
```

## Champs de données

- [Town firstTown](#)  
*Première ville.*
- [Town secondTown](#)  
*Seconde ville.*
- double [distance](#)  
*Distance entre les deux villes.*

### 4.2.1 Description détaillée

Objet des distances. [Distance](#) entre deux villes

#### Voir également

[Town](#)

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

- lib/[distance.h](#)

## 4.3 Référence de la structure Errors

Objet des erreurs.

```
#include <errors.h>
```

## Champs de données

- char \* [errorNbArguments](#)  
*Nombre d'argument incorrect.*
- char \* [errorTagNotFound](#)  
*Tag -f non trouvé.*
- char \* [errorFileNotFound](#)  
*Fichier non trouvé.*
- char \* [errorNoAlgoSpecified](#)  
*Algorithme non spécifié.*
- char \* [errorMissingParameterLsr](#)  
*Paramètre après -lsr manquant.*
- char \* [errorMissingParameterLsnr](#)  
*Paramètre après -lsnr manquant.*
- char \* [errorMissingParameterGa](#)  
*Paramètre après -ga manquant.*
- char \* [errorNoValidParameterLsr](#)  
*Paramètre après -lsr non valide.*



- char \* [errorNoValidParameterLsnr](#)  
*Paramètre après -lsnr non valide.*
- char \* [errorNoValidParameterGa](#)  
*Paramètre après -ga non valide.*
- int [nbErrors](#)  
*Nombre d'erreurs.*

### 4.3.1 Description détaillée

Objet des erreurs. Toutes les chaînes de caractères des erreurs. Si une variable vaut NULL, l'erreur n'est pas présente, sinon elle sera affichée. nbErrors est le nombre d'erreur, si celui-ci =0 alors le programme peut fonctionner correctement

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

- lib/[errors.h](#)

## 4.4 Référence de la structure Instance

Objet des instances.

```
#include <instance.h>
```

### Champs de données

- [Town towns](#) [N]  
*Tableau des villes( Town ) classés par ID.*
- [Distance distances](#) [N]  
*Tableau linéaire contenant toutes les distances entre les villes.*
- int [nbTowns](#)  
*Nombre de ville de l' [Instance](#).*
- char \* [name](#)  
*Nom de l' [Instance](#).*
- char \* [type](#)  
*Type de l' [Instance](#).*

### 4.4.1 Description détaillée

Objet des instances.

**Voir également**

[Town](#)  
[Distance](#)

Une instance contient toutes les villes ( [Town](#) ) classés par ID, les calculs des algorithmes utilisent une instance afin d'en retourner la meilleur tournée.

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

- lib/[instance.h](#)

## 4.5 Référence de la structure Tour

Objet d'une tournée.

```
#include <tour.h>
```

### Champs de données

- [Town towns](#) [N]  
*Tableau de ville. Les villes sont triés dans l'ordre de la tournée.*
- int [nbTowns](#)  
*Nombre de ville de la tournée.*
- double [length](#)  
*Longueur de la tournée.*
- [Distance](#) \* [distances](#)  
*Matrice de distances.*

### 4.5.1 Description détaillée

Objet d'une tournée. Informations concernant une tournée.

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

- lib/[tour.h](#)

## 4.6 Référence de la structure Town

Objet des ville.

```
#include <town.h>
```

### Champs de données

- int [x](#)  
*Abscisse de la ville.*
- int [y](#)  
*Ordonnée de la ville.*
- int [id](#)  
*Id de la ville.*

### **4.6.1 Description détaillée**

Objet des ville. Structure représentant une ville

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

- lib/[town.h](#)



# Chapitre 5

## Documentation des fichiers

### 5.1 Référence du fichier lib/bruteForce.h

Fonctions de brute force.

```
#include "instance.h"
#include "tour.h"
#include "util.h"
```

#### Fonctions

- [Tour bruteForce\\_bestPath](#) ([Instance](#) pInstance)  
*Permet d'obtenir le meilleur chemin d'une instance via la force brute.*

#### 5.1.1 Description détaillée

Fonctions de brute force.

##### Auteur

Antoine de Roquemaurel

##### Date

27/12/2012 17 :58 :36

Entêtes des fonctions servant à la force brute. Ces fonctions ne sont appelés que depuis le main

#### 5.1.2 Documentation des fonctions

##### 5.1.2.1 Tour bruteForce\_bestPath ( Instance *pInstance* )

Permet d'obtenir le meilleur chemin d'une instance via la force brute.

Il est conseillé de ne pas essayer avec des instances de plus de 8 villes.

##### Paramètres

*pInstance* L'instance pour laquelle on doit calculer le plus court chemin

**Renvoi**

La meilleur tournée.

## 5.2 Référence du fichier lib/distance.h

Fonctions des distances.

```
#include <math.h>
#include "town.h"
```

**Structures de données**

- struct [Distance](#)  
*Objet des distances.*

**Fonctions**

- [Distance](#) [distance\\_new](#) ([Town](#) \*pFirstTown, [Town](#) \*pSecondTown)  
*Créer une nouvelle distance.*
- double [distance\\_calculDistance](#) (const [Town](#) pTown1, const [Town](#) pTown2)  
*Calcul la distance entre deux villes.*
- double [distance\\_betweenTowns](#) ([Distance](#) \*pDistances, int i, int j)  
*Calcul la distance entre deux ID de villes.*
- [Distance](#) [distance\\_searchDistance](#) ([Distance](#) \*pDistances, const int pFirst, const int pSecond)  
*Recherche la distance entre deux villes dans le tableau de distance.*

### 5.2.1 Description détaillée

Fonctions des distances.

**Auteur**

Antoine de Roquemaurel

**Date**

01/12/2012 20 :33 :44

Entêtes des fonctions se rapportant aux distances

### 5.2.2 Documentation des fonctions

#### 5.2.2.1 double [distance\\_betweenTowns](#) ( [Distance](#) \* *pDistances*, int *i*, int *j* )

Calcul la distance entre deux ID de villes.

**Paramètres**

*pDistances* La matrice de distances

*i* La première ville

*j* La seconde ville

**Renvoie**

La distance

**5.2.2.2 double distance\_calculDistance ( const Town *pTown1*, const Town *pTown2* )**

Calcul la distance entre deux villes.

**Paramètres**

*pTown1* Première ville

*pTown2* Seconde ville

**Renvoie**

La distance

**5.2.2.3 Distance distance\_new ( Town \* *pFirstTown*, Town \* *pSecondTown* )**

Créer une nouvelle distance.

**Paramètres**

*pFirstTown* Première ville

*pSecondTown* Seconde ville

**Renvoie**

[Distance](#) entre les deux villes

**5.2.2.4 Distance distance\_searchDistance ( Distance \* *pDistances*, const int *pFirst*, const int *pSecond* )**

Recherche la distance entre deux villes dans le tableau de distance.

**Paramètres**

*pDistances* La matrice de distance

*pFirst* La première ville

*pSecond* la seconde ville

**Renvoie**

La distance

## 5.3 Référence du fichier lib/errors.h

Fonctions sur les erreurs.

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

```
#include <stdlib.h>
```

## Structures de données

- struct `Errors`  
*Objet des erreurs.*

## Fonctions

- `Errors errors_new ()`  
*Initialisation de l'objet.*
- void `errors_displayErrorMessage (const Errors pErrors)`  
*Affiches toutes les erreurs de pErrors.*
- void `errors_setNbArguments (Errors *pErrors)`  
*Signale que le nombre d'argument est incorrect.*
- void `errors_setTagFNotFound (Errors *pErrors)`  
*Signale que la balise -f n'a pas été trouvée.*
- void `errors_setFileNotFound (Errors *pErrors, char *fileName)`  
*Signale que le fichier fileName n'existe pas.*
- void `errors_setNoAlgoSpecified (Errors *pErrors)`  
*Signale qu'aucun algorithme n'a été spécifié.*
- void `errors_setNoValidParameterLsnr (Errors *pErrors)`  
*Signale que les paramètres derrière -lsnr ne sont pas valides (non entier).*
- void `errors_setNoValidParameterLsr (Errors *pErrors)`  
*Signale que le paramètre derrière -lsr n'est pas valide (n'est pas un entier).*
- void `errors_setMissingParameterGa (Errors *pErrors)`  
*Signale qu'aucun paramètre n'a été spécifié derrière -ga.*
- void `errors_setMissingParameterLsnr (Errors *pErrors)`  
*Signale qu'aucun paramètre n'a été spécifié derrière -lsnr.*
- void `errors_setMissingParameterLsr (Errors *pErrors)`  
*Signale qu'aucun paramètre n'a été spécifié derrière -lsr.*

### 5.3.1 Description détaillée

Fonctions sur les erreurs.

#### Auteur

Antoine de Roquemaurel

#### Date

21/11/2012 17 :42 :37

Entêtes des fonctions concernant les erreurs du programme.



## 5.3.2 Documentation des fonctions

### 5.3.2.1 void errors\_displayErrorsMessage ( const Errors *pErrors* )

Affiches toutes les erreurs de *pErrors*.

#### Paramètres

*pErrors* L'objet pour laquelle on doit afficher les erreurs

### 5.3.2.2 Errors errors\_new ( )

Initialisation de l'objet.

#### Renvoie

Une instance de Erreur initialisée

### 5.3.2.3 void errors\_setFileNotFound ( Errors \* *pErrors*, char \* *fileName* )

Signale que le fichier *fileName* n'existe pas.

#### Paramètres

*pErrors* L'objet des erreurs

*fileName* Le nom de fichier non trouvé

### 5.3.2.4 void errors\_setMissingParameterGa ( Errors \* *pErrors* )

Signale qu'aucun paramètre n'a été spécifié derrière -ga.

#### Paramètres

*pErrors* L'objet des erreurs

### 5.3.2.5 void errors\_setMissingParameterLsnr ( Errors \* *pErrors* )

Signale qu'aucun paramètre n'a été spécifié derrière -lsnr.

#### Paramètres

*pErrors* L'objet des erreurs

### 5.3.2.6 void errors\_setMissingParameterLsr ( Errors \* *pErrors* )

Signale qu'aucun paramètre n'a été spécifié derrière -lsr.

#### Paramètres

*pErrors* L'objet des erreurs

### 5.3.2.7 void errors\_setNbArguments ( Errors \* *pErrors* )

Signale que le nombre d'argument est incorrect.

#### Paramètres

*pErrors* L'objet des erreurs

### 5.3.2.8 void errors\_setNoAlgoSpecified ( Errors \* *pErrors* )

Signale qu'aucun algorithme n'a été spécifié.

#### Paramètres

*pErrors* L'objet des erreurs

### 5.3.2.9 void errors\_setNoValidParameterLsnr ( Errors \* *pErrors* )

Signale que les paramètres derrière -lsnr ne sont pas valides (non entier).

#### Paramètres

*pErrors* L'objet des erreurs

### 5.3.2.10 void errors\_setNoValidParameterLsr ( Errors \* *pErrors* )

Signale que le paramètre derrière -lsr n'est pas valide (n'est pas un entier).

#### Paramètres

*pErrors* L'objet des erreurs

### 5.3.2.11 void errors\_setTagFNotFound ( Errors \* *pErrors* )

Signale que la balise -f n'a pas été trouvée.

#### Paramètres

*pErrors* L'objet des erreurs

## 5.4 Référence du fichier lib/instance.h

Fonctions sur les instances.

```
#include <stdbool.h>
#include <string.h>
#include "town.h"
#include "distance.h"
```

## Structures de données

- struct `Instance`  
*Objet des instances.*

## Macros

- #define `N` 1024  
*Taille maximale des tableaux.*

## Fonctions

- void `instance_display` (const `Instance` `pInstance`)  
*Affiche une `Instance`.*
- `Instance` `instance_new` (FILE `*pFile`)  
*Créer une nouvelle `Instance` à partir d'un fichier.*
- void `instance_push` (`Instance` `*pInstance`, const `Town` `pTown`)  
*Ajoute une nouvelle ville dans une `Instance`.*
- void `instance_initializeDistancesMatrix` (`Instance` `*pInstance`)  
*Initialise la matrice des distances.*
- void `instance_displayLinearVector` (`Instance` `pInstance`)  
*Affiche la matrice des distances sous forme linéaire.*
- void `instance_displayMatrix` (`Instance` `pInstance`)  
*Affiche la matrice des distances sous forme de matrice symétrique.*

### 5.4.1 Description détaillée

Fonctions sur les instances.

#### Auteur

Antoine de Roquemaurel

#### Date

21/11/2012 22 :03 :34

Toutes les entêtes des fonctions se rapportant à une instance.

### 5.4.2 Documentation des fonctions

#### 5.4.2.1 void `instance_display` ( const `Instance` `pInstance` )

Affiche une `Instance`.

#### Paramètres

`pInstance` l' `Instance` à afficher

#### 5.4.2.2 void instance\_displayLinearVector ( Instance *pInstance* )

Affiche la matrice des distances sous forme linéaire.

##### Paramètres

*pInstance* L'instance à afficher

##### Voir également

[Distance](#)

#### 5.4.2.3 void instance\_displayMatrix ( Instance *pInstance* )

Affiche la matrice des distances sous forme de matrice symétrique.

##### Paramètres

*pInstance* L'instance à afficher

##### Voir également

[Distance](#)

#### 5.4.2.4 void instance\_initializeDistancesMatrix ( Instance \* *pInstance* )

Initialise la matrice des distances.

##### Paramètres

*pInstance* L'instance à modifier

##### Voir également

[Distance](#)

#### 5.4.2.5 Instance instance\_new ( FILE \* *pFile* )

Créer une nouvelle [Instance](#) à partir d'un fichier.

##### Paramètres

*pFile* Le pointeur sur fichier contenant les informations de l'instance

##### Renvoie

la nouvelle [Instance](#)

#### 5.4.2.6 void instance\_push ( Instance \* *pInstance*, const Town *pTown* )

Ajoute une nouvelle ville dans une [Instance](#).

##### Paramètres

*pInstance* L'instance à modifier

*pTown* La ville à ajouter

## 5.5 Référence du fichier lib/localSearch.h

Fonctions utiles.

```
#include "tour.h"
#include <time.h>
#include <stdlib.h>
```

### Fonctions

- [Tour](#) `localSearch_randomBestPath` ([Instance](#) pInstance, int pTryNb)
- [Tour](#) `localSearch_systematicBestPath` ([Instance](#) pInstance, int pTryNb)

### 5.5.1 Description détaillée

Fonctions utiles. Fonctions de recherche locale.

#### Auteur

Antoine de Roquemaurel

#### Date

27/12/2012 18 :00 :30

Entêtes des fonctions pouvant être utiles dans tout le projet. Ces fonctions ne sont appelés que depuis le main

#### Auteur

Antoine de Roquemaurel

#### Date

27/12/2012 18 :00 :18

Entêtes des fonctions ayant rapport avec les algorithmes de recherches locales.

## 5.6 Référence du fichier lib/parsing.h

Fonctions de parsing des arguments.

```
#include <stdbool.h>
#include "util.h"
#include "errors.h"
```

### Structures de données

- struct [Algo](#)  
*Enumération d'un algorithme.*

## Énumérations

- enum [AlgoType](#) {  
    [BRUTEFORCE](#), [LOCALSEARCH\\_RANDOM](#), [LOCALSEARCH\\_SYSTEMATIC](#), [GENETIC](#),  
    [END](#) }  
    [parsing.h](#)

## Fonctions

- bool [parsing\\_parseVerboseMode](#) (char \*\*pTab, const int pSize)  
    *Cherche si le mode verbeux à été spécifié ou non.*
- char \* [parsing\\_parseFileName](#) (char \*\*pTab, const int pSize, [Errors](#) \*pErrors)  
    *Cherche le nom du fichier de l'Instance.*
- void [parsing\\_algoType](#) (char \*\*pTab, const int pSize, [Errors](#) \*pErrors, [Algo](#) \*algos)

### 5.6.1 Description détaillée

Fonctions de parsing des arguments.

#### Auteur

Antoine de Roquemaurel

#### Date

21/11/2012 17 :17 :24

Entêtes des fonctions permettant de parser les arguments, et ainsi d'appeler les différents algorithmes demandés, d'utiliser le mode verbeux et de spécifier le fichier.

### 5.6.2 Documentation du type de l'énumération

#### 5.6.2.1 enum AlgoType

[parsing.h](#)

Énumération des types d'algorithmes Les différents types d'algorithmes qui peuvent être appelés.

#### Valeurs énumérées :

***BRUTEFORCE*** L'algorithme de brute force.  
***LOCALSEARCH\_RANDOM*** L'algorithme de recherche locale aléatoire.  
***LOCALSEARCH\_SYSTEMATIC*** L'algorithme de recherche locale systématique.  
***GENETIC*** L'algorithme génétique.  
***END*** Correspond au marqueur de fin des algorithmes.

### 5.6.3 Documentation des fonctions

#### 5.6.3.1 void parsing\_algoType ( char \*\* pTab, const int pSize, Errors \* pErrors, Algo \* algos )

#### Paramètres

***pTab*** Le tableau contenant les arguments

*pSize* Le nombre des arguments

*pErrors* L'objet des erreurs, il est modifié si des erreurs interviennent

*algos* Tableau d'algorithmes, ceci au cas où l'utilisateur entre plusieurs algorithmes. La fin du tableau est marqué par END

#### Voir également

[Algo](#)

#### 5.6.3.2 char\* parsing\_parseFileName ( char \*\* *pTab*, const int *pSize*, Errors \* *pErrors* )

Cherche le nom du fichier de l'Instance.

##### Paramètres

*pTab* Le tableau contenant les arguments

*pSize* Le nombre des arguments

*pErrors* L'objet des erreurs, il est modifié si des erreurs interviennent

##### Renvoie

Le nom du fichier

#### Voir également

[Instance](#)

#### 5.6.3.3 bool parsing\_parseVerboseMode ( char \*\* *pTab*, const int *pSize* )

Cherche si le mode verbeux a été spécifié ou non.

##### Paramètres

*pTab* Le tableau contenant les arguments

*pSize* Le nombre des arguments

##### Renvoie

Vrai si mode verbeux, Faux sinon

## 5.7 Référence du fichier lib/tour.h

Fonctions des tournées.

```
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include "town.h"
#include "instance.h"
#include "distance.h"
```

## Structures de données

- struct `Tour`  
*Objet d'une tournée.*

## Fonctions

- `Tour tour_new (Instance pInstance)`  
*Créer une nouvelle tournée initialisée avec les données d'une instance.*
- `bool tour_nextPermutation (Tour *pPermutation)`  
*Génère la permutation de ville suivante d'une tournée.*
- `void tour_calculLength (Tour *pTour)`  
*Calcul la longueur d'une tournée.*
- `void tour_display (const Tour pTour)`  
*Affiche une tournée.*
- `Tour tour_randomWalk (const Instance pInstance)`  
*Génère une tournée aléatoire.*
- `void tour_2opt (Tour *pTour, int pFirst, int pSecond)`  
*Fait une 2opt.*

### 5.7.1 Description détaillée

Fonctions des tournées.

#### Auteur

Antoine de Roquemaurel

#### Date

21/11/2012 22 :04 :13

Entêtes des fonctions se rapportant à une tournée. \*

### 5.7.2 Documentation des fonctions

#### 5.7.2.1 void tour\_2opt ( Tour \* pTour, int pFirst, int pSecond )

Fait une 2opt.

#### Paramètres

*pTour* Le tour pour laquelle on veut faire une 2opt

*pFirst* L'id du début du premier trajet

*pSecond* L'id du début du second trajet

#### 5.7.2.2 void tour\_calculLength ( Tour \* pTour )

Calcul la longueur d'une tournée.



**Paramètres**

*pTour* La tournée pour laquelle on veut calculer la longueur

**5.7.2.3 void tour\_display ( const Tour pTour )**

Affiche une tournée.

**Paramètres**

*pTour* La tournée à afficher

**5.7.2.4 Tour tour\_new ( Instance pInstance )**

Créer une nouvelle tournée initialisée avec les données d'une instance.

**Paramètres**

*pInstance* Instance servant à initialiser la tournée

**Renvoie**

la nouvelle tournée

**5.7.2.5 bool tour\_nextPermutation ( Tour \* pPermutation )**

Génère la permutation de ville suivante d'une tournée.

**Paramètres**

*pPermutation* La tournée pour laquelle la permutation doit être générée

**Renvoie**

Vrai si une permutation a été générée faux s'il ne reste plus de permutation.

**5.7.2.6 Tour tour\_randomWalk ( const Instance pInstance )**

Génère une tournée aléatoire.

**Paramètres**

*pInstance* L'instance pour laquelle générer un random walk

**Renvoie**

La tournée aléatoire

## 5.8 Référence du fichier lib/town.h

Fonctions des villes.

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

```
#include <stdlib.h>
```

## Structures de données

- struct [Town](#)  
*Objet des ville.*

## Fonctions

- [Town town\\_new](#) (const int pId, const int pX, const int pY)  
*Création d'une nouvelle ville.*

### 5.8.1 Description détaillée

Fonctions des villes.

#### Auteur

Antoine de Roquemaurel

#### Date

21/11/2012 22 :35 :19

Entêtes des fonctions se rapportant à une ville.

### 5.8.2 Documentation des fonctions

#### 5.8.2.1 Town town\_new ( const int *pId*, const int *pX*, const int *pY* )

Création d'une nouvelle ville.

#### Paramètres

*pId* Id de la ville

*pX* Abscisse

*pY* Ordonnée

## 5.9 Référence du fichier lib/util.h

Fonctions utiles.

```
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
#include "tour.h"
```

## Fonctions

- int [util\\_searchFirstOccurenceInArray](#) (char \*\*pArray, const int pSize, char \*pSearch)  
*Cherche la première occurrence d'une chaîne de caractère dans un tableau de chaîne caractère.*

- void `util_reverseArray` (`Town *pTab`, const int pBegin, const int pEnd)  
*Inverse les éléments d'un tableau pTab entre les cases pBegin et pEnd.*
- void `util_displayArray` (const int \*pTab, const int pSize)  
*Affiche le contenu d'un tableau d'entiers.*
- int `util_sum` (const int pBegin, const int pEnd)  
*Calcul la somme des éléments allant de pBegin à pEnd.*
- void `util_swap` (int \*a, int \*b)  
*Échange deux variables.*
- int `util_rand` (const int pMin, const int pMax)  
*Calcul une valeur aléatoire entre pMin et pMax.*
- bool `util_sousTabExist` (`Tour pChild`, const int pBegin, const int pEnd, `Tour pParent`, bool pRecursvite)  
*Indique si un sous-tableau d'un tableau est présent dans un autre tableau.*

## Variables

- bool `gVerboseMode`  
*Mode verbose.*

### 5.9.1 Description détaillée

Fonctions utiles.

#### Auteur

Antoine de Roquemaurel

#### Date

19/11/2012 16 :27 :39

Entêtes des fonctions pouvant être utiles dans tout le projet. Ce sont des fonctions simples, qui doivent être indépendantes du projet.

### 5.9.2 Documentation des fonctions

#### 5.9.2.1 void util\_displayArray ( const int \* pTab, const int pSize )

Affiche le contenu d'un tableau d'entiers.

##### Paramètres

*pTab* Le tableau à afficher

*pSize* La taille du tableau

#### 5.9.2.2 int util\_rand ( const int pMin, const int pMax )

Calcul une valeur aléatoire entre pMin et pMax.

**Paramètres**

*pMin* Minimum

*pMax* Maximum

**Renvoie****5.9.2.3 void util\_reverseArray ( Town \* pTab, const int pBegin, const int pEnd )**

Inverse les éléments d'un tableau pTab entre les cases pBegin et pEnd.

**Paramètres**

*pTab* Tableau à inverser

*pBegin* Début de la section à inverser

*pEnd* Fin de la section à inverser

**5.9.2.4 int util\_searchFirstOccurenceInArray ( char \*\* pArray, const int pSize, char \* pSearch )**

Cherche la première occurrence d'une chaîne de caractère dans un tableau de chaînes de caractères.

**Paramètres**

*pArray* Le tableau de chaînes de caractères dans lequel chercher

*pSize* La taille du tableau

*pSearch* La chaîne de caractère à chercher

**Renvoie**

La position de la chaîne dans le tableau ou -1 si elle n'a pas été trouvée

**5.9.2.5 bool util\_sousTabExist ( Tour pChild, const int pBegin, const int pEnd, Tour pParent, bool pRecursive )**

Indique si un sous-tableau d'un tableau est présent dans un autre tableau.

**Paramètres**

*pChild* Le tableau à chercher

*pBegin* Le début de la section à chercher

*pEnd* La fin de la section à chercher

*pParent* Le tableau dans lequel chercher

*pRecursive* Pour simplifier vis-à-vis de la récursivité, doit toujours être à false

**Renvoie**

Vrai si le sous-tableau a été trouvé faux sinon

### 5.9.2.6 int util\_sum ( const int *pBegin*, const int *pEnd* )

Calcul la somme des éléments allant de *pBegin* à *pEnd*.

#### Paramètres

*pBegin* Début de la somme

*pEnd* Fin de la somme

#### Renvoie

La somme des éléments

### 5.9.2.7 void util\_swap ( int \* *a*, int \* *b* )

Échange deux variables.

#### Paramètres

*a* Première variable à échanger

*b* Seconde variable

## 5.10 Référence du fichier src/bruteForce.c

Fonctions de brute force.

```
#include "bruteForce.h"
```

```
#include "util.h"
```

### Fonctions

- [Tour bruteForce\\_bestPath](#) (Instance *pInstance*)  
*Permet d'obtenir le meilleur chemin d'une instance via la force brute.*

### 5.10.1 Description détaillée

Fonctions de brute force. Fonctions sur les erreurs.

#### Auteur

Antoine de Roquemaurel

#### Date

27/12/2012 17 :58 :29

Implémentation des fonctions utilisant la force brute. Ces fonctions ne sont appelés que depuis le main

#### Auteur

Antoine de Roquemaurel

**Date**

21/11/2012 17 :42 :31

Implémentation des fonctions concernant les erreurs du programme.

**5.10.2 Documentation des fonctions****5.10.2.1 Tour bruteForce\_bestPath ( Instance *pInstance* )**

Permet d'obtenir le meilleur chemin d'une instance via la force brute.

Il est conseillé de ne pas essayer avec des instances de plus de 8 villes.

**Paramètres**

*pInstance* L'instance pour laquelle on doit calculer le plus court chemin

**Renvoie**

La meilleur tournée.

**5.11 Référence du fichier src/distance.c**

Fonctions des distances.

```
#include "distance.h"
```

```
#include "util.h"
```

**Fonctions**

- `Distance distance_new (Town *pFirstTown, Town *pSecondTown)`  
*Créer une nouvelle distance.*
- `double distance_calculDistance (const Town pTown1, const Town pTown2)`  
*Calcul la distance entre deux villes.*
- `Distance distance_searchDistance (Distance *pDistances, const int pFirst, const int pSecond)`  
*Recherche la distance entre deux villes dans le tableau de distance.*
- `double distance_betweenTowns (Distance *pDistances, int i, int j)`  
*Calcul la distance entre deux ID de villes.*

**5.11.1 Description détaillée**

Fonctions des distances.

**Auteur**

Antoine de Roquemaurel

**Date**

01/12/2012 20 :33 :39

Entêtes des fonctions se rapportant aux distances

## 5.11.2 Documentation des fonctions

### 5.11.2.1 double distance\_betweenTowns ( Distance \* *pDistances*, int *i*, int *j* )

Calcul la distance entre deux ID de villes.

#### Paramètres

*pDistances* La matrice de distances

*i* La première ville

*j* La seconde ville

#### Renvoie

La distance

### 5.11.2.2 double distance\_calculDistance ( const Town *pTown1*, const Town *pTown2* )

Calcul la distance entre deux villes.

#### Paramètres

*pTown1* Première ville

*pTown2* Seconde ville

#### Renvoie

La distance

### 5.11.2.3 Distance distance\_new ( Town \* *pFirstTown*, Town \* *pSecondTown* )

Créer une nouvelle distance.

#### Paramètres

*pFirstTown* Première ville

*pSecondTown* Seconde ville

#### Renvoie

[Distance](#) entre les deux villes

### 5.11.2.4 Distance distance\_searchDistance ( Distance \* *pDistances*, const int *pFirst*, const int *pSecond* )

Recherche la distance entre deux villes dans le tableau de distance.

#### Paramètres

*pDistances* La matrice de distance

*pFirst* La première ville

*pSecond* la seconde ville

#### Renvoie

La distance

## 5.12 Référence du fichier src/genetic.c

Fonctions d'algorithmes génétiques.

```
#include "genetic.h"
#include "tour.h"
```

### Fonctions

- [Tour](#) `genetic_distancePreservingCrossover` ([Tour](#) pParent1, [Tour](#) pParent2)

### 5.12.1 Description détaillée

Fonctions d'algorithmes génétiques.

#### Auteur

Antoine de Roquemaurel

#### Date

27/12/2012 18 :00 :25

Implémentation des fonctions servant aux algorithmes génétiques. Ces fonctions ne sont appelés que depuis le main

## 5.13 Référence du fichier src/instance.c

Fonctions utiles.

```
#include "instance.h"
```

### Fonctions

- [Instance](#) `instance_new` (FILE \*pFile)  
*Créer une nouvelle [Instance](#) à partir d'un fichier.*
- void `instance_display` (const [Instance](#) pInstance)  
*Affiche une [Instance](#).*
- void `instance_push` ([Instance](#) \*pInstance, const [Town](#) pTown)  
*Ajoute une nouvelle ville dans une [Instance](#).*
- void `instance_initializeDistancesMatrix` ([Instance](#) \*pInstance)  
*Initialise la matrice des distances.*
- void `instance_displayLinearVector` ([Instance](#) pInstance)  
*Affiche la matrice des distances sous forme linéaire.*
- void `instance_displayMatrix` ([Instance](#) pInstance)  
*Affiche la matrice des distances sous forme de matrice symétrique.*



### 5.13.1 Description détaillée

Fonctions utiles.

**Auteur**

Antoine de Roquemaurel

**Date**

21/11/2012 22 :03 :26

Entêtes des fonctions pouvant être utiles dans tout le projet. Ce sont des fonctions simples, qui doivent être indépendantes du projet.

### 5.13.2 Documentation des fonctions

#### 5.13.2.1 void instance\_display ( const Instance *pInstance* )

Affiche une [Instance](#).

**Paramètres**

*pInstance* l' [Instance](#) à afficher

#### 5.13.2.2 void instance\_displayLinearVector ( Instance *pInstance* )

Affiche la matrice des distances sous forme linéaire.

**Paramètres**

*pInstance* L'instance à afficher

**Voir également**

[Distance](#)

#### 5.13.2.3 void instance\_displayMatrix ( Instance *pInstance* )

Affiche la matrice des distances sous forme de matrice symétrique.

**Paramètres**

*pInstance* L'instance à afficher

**Voir également**

[Distance](#)

#### 5.13.2.4 void instance\_initializeDistancesMatrix ( Instance \* *pInstance* )

Initialise la matrice des distances.

**Paramètres**

*pInstance* L'instance à modifier

Voir également

[Distance](#)

#### 5.13.2.5 Instance `instance_new ( FILE * pFile )`

Cr  er une nouvelle [Instance](#)    partir d'un fichier.

**Param  tres**

*pFile* Le pointeur sur fichier contenant les informations de l'instance

**Renvoie**

la nouvelle [Instance](#)

#### 5.13.2.6 `void instance_push ( Instance * pInstance, const Town pTown )`

Ajoute une nouvelle ville dans une [Instance](#).

**Param  tres**

*pInstance* L'instance    modifier

*pTown* La ville    ajouter

## 5.14 R  f  rence du fichier `src/localSearch.c`

Fonctions de recherche locale.

```
#include "util.h"
#include "localSearch.h"
#include "instance.h"
#include "tour.h"
```

**Fonctions**

- [Tour](#) `localSearch_randomBestPath (Instance pInstance, int pTryNb)`
- [Tour](#) `localSearch_systematicBestPath (Instance pInstance, int pTryNb)`

### 5.14.1 Description d  taill  e

Fonctions de recherche locale.

**Auteur**

Antoine de Roquemaurel

**Date**

27/12/2012 18 :00 :13

Ent  tes des fonctions ayant rapport avec les algorithmes de recherches locales.

## 5.15 Référence du fichier src/main.c

Fonction main.

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include "parsing.h"
#include "errors.h"
#include "bruteForce.h"
#include "localSearch.h"
```

### Fonctions

- int `main` (int argc, char \*\*argv)  
*Fonction d'entrée du programme.*

### 5.15.1 Description détaillée

Fonction main.

#### Auteur

Antoine de Roquemaurel

#### Date

19/11/2012 10 :42 :29

Point d'entrée du programme. Aucune fonction ne doit être déclarée Ce sont des fonctions simples, qui doivent être indépendantes du projet.

### 5.15.2 Documentation des fonctions

#### 5.15.2.1 int main ( int argc, char \*\* argv )

Fonction d'entrée du programme.

#### Paramètres

*argc* Nombre d'arguments du programme

*argv* Tableau contenant la liste des arguments du programme

#### Renvoie

Code d'erreur du programme

## 5.16 Référence du fichier src/parsing.c

Fonctions de parsing des arguments.

```
#include "parsing.h"
```

## Fonctions

- bool [parsing\\_parseVerboseMode](#) (char \*\*pTab, const int pSize)  
*Cherche si le mode verbeux à été spécifié ou non.*
- char \* [parsing\\_parseFileName](#) (char \*\*pTab, const int pSize, [Errors](#) \*pErrors)  
*Cherche le nom du fichier de l'Instance.*
- void [parsing\\_algoType](#) (char \*\*pTab, const int pSize, [Errors](#) \*pErrors, [Algo](#) \*algos)

### 5.16.1 Description détaillée

Fonctions de parsing des arguments.

#### Auteur

Antoine de Roquemaurel

#### Date

21/11/2012 17 :17 :18

Implémentation des fonctions permettant de parser les arguments, et ainsi d'appeler les différents algorithmes demandés, d'utiliser le mode verbeux et de spécifier le fichier.

### 5.16.2 Documentation des fonctions

#### 5.16.2.1 void [parsing\\_algoType](#) ( char \*\* *pTab*, const int *pSize*, [Errors](#) \* *pErrors*, [Algo](#) \* *algos* )

##### Paramètres

*pTab* Le tableau contenant les arguments

*pSize* Le nombre des arguments

*pErrors* L'objet des erreurs, il est modifié si des erreurs interviennent

*algos* Tableau d'algorithmes, ceci au cas où l'utilisateur entre plusieurs algorithmes. La fin du tableau est marqué par END

##### Voir également

[Algo](#)

#### 5.16.2.2 char\* [parsing\\_parseFileName](#) ( char \*\* *pTab*, const int *pSize*, [Errors](#) \* *pErrors* )

Cherche le nom du fichier de l'Instance.

##### Paramètres

*pTab* Le tableau contenant les arguments

*pSize* Le nombre des arguments

*pErrors* L'objet des erreurs, il est modifié si des erreurs interviennent

##### Renvoie

Le nom du fichier

##### Voir également

[Instance](#)

### 5.16.2.3 bool parsing\_parseVerboseMode ( char \*\* pTab, const int pSize )

Cherche si le mode verbeux à été spécifié ou non.

#### Paramètres

*pTab* Le tableau contenant les arguments

*pSize* Le nombre des arguments

#### Renvoie

Vrai si mode verbeux, Faux sinon

## 5.17 Référence du fichier src/tour.c

Fonctions des tournées.

```
#include "tour.h"
```

### Fonctions

- [Tour](#) [tour\\_new](#) ([Instance](#) pInstance)  
*Créer une nouvelle tournée initialisée avec les données d'une instance.*
- bool [tour\\_nextPermutation](#) ([Tour](#) \*pPermutation)  
*Génère la permutation de ville suivante d'une tournée.*
- void [tour\\_calculLength](#) ([Tour](#) \*pTour)  
*Calcul la longueur d'une tournée.*
- void [tour\\_display](#) (const [Tour](#) pTour)  
*Affiche une tournée.*
- [Tour](#) [tour\\_randomWalk](#) (const [Instance](#) pInstance)  
*Génère une tournée aléatoire.*
- void [tour\\_2opt](#) ([Tour](#) \*pTour, int pFirst, int pSecond)  
*Fait une 2opt.*

### 5.17.1 Description détaillée

Fonctions des tournées.

#### Auteur

Antoine de Roquemaurel

#### Date

21/11/2012 22 :04 :08

Implémentation des fonctions se rapportant à une tournée.

## 5.17.2 Documentation des fonctions

### 5.17.2.1 void tour\_2opt ( Tour \* *pTour*, int *pFirst*, int *pSecond* )

Fait une 2opt.

#### Paramètres

*pTour* Le tour pour laquelle on veut faire une 2opt

*pFirst* L'id du début du premier trajet

*pSecond* L'id du début du second trajet

### 5.17.2.2 void tour\_calculLength ( Tour \* *pTour* )

Calcul la longueur d'une tournée.

#### Paramètres

*pTour* La tournée pour laquelle on veut calculer la longueur

### 5.17.2.3 void tour\_display ( const Tour *pTour* )

Affiche une tournée.

#### Paramètres

*pTour* La tournée à afficher

### 5.17.2.4 Tour tour\_new ( Instance *pInstance* )

Créer une nouvelle tournée initialisée avec les données d'une instance.

#### Paramètres

*pInstance* [Instance](#) servant à initialiser la tournée

#### Renvoie

la nouvelle tournée

### 5.17.2.5 bool tour\_nextPermutation ( Tour \* *pPermutation* )

Génère la permutation de ville suivante d'une tournée.

#### Paramètres

*pPermutation* La tournée pour laquelle la permutation doit être générée

#### Renvoie

Vrai si une permutation a été générée faux s'il ne reste plus de permutation.

### 5.17.2.6 Tour `tour_randomWalk ( const Instance pInstance )`

Génère une tournée aléatoire.

#### Paramètres

*pInstance* L'instance pour laquelle générer un random walk

#### Renvoie

La tournée aléatoire

## 5.18 Référence du fichier src/town.c

Fonctions des villes.

```
#include "town.h"
```

### Fonctions

- [Town `town\_new`](#) (const int pId, const int pX, const int pY)  
*Création d'une nouvelle ville.*

### 5.18.1 Description détaillée

Fonctions des villes.

#### Auteur

Antoine de Roquemaurel

#### Date

21/11/2012 22 :35 :14

implémentation des fonctions se rapportant à une ville.

### 5.18.2 Documentation des fonctions

#### 5.18.2.1 Town `town_new ( const int pId, const int pX, const int pY )`

Création d'une nouvelle ville.

#### Paramètres

*pId* Id de la ville

*pX* Abscisse

*pY* Ordonnée

## 5.19 Référence du fichier src/util.c

Fonctions utiles.

```
#include "tour.h"
#include "util.h"
```

## Fonctions

- int [util\\_searchFirstOccurenceInArray](#) (char \*\*pArray, const int pSize, char \*pSearch)  
*Cherche la première occurrence d'une chaîne de caractère dans un tableau de chaîne caractère.*
- void [util\\_reverseArray](#) (Town \*pTab, const int pBegin, const int pEnd)  
*Inverse les éléments d'un tableau pTab entre les cases pBegin et pEnd.*
- void [util\\_displayArray](#) (const int \*pTab, const int pSize)  
*Affiche le contenu d'un tableau d'entiers.*
- int [util\\_sum](#) (const int pBegin, const int pEnd)  
*Calcul la somme des éléments allant de pBegin à pEnd.*
- void [util\\_swap](#) (int \*a, int \*b)  
*Échange deux variables.*
- int [util\\_rand](#) (const int pMin, const int pMax)  
*Calcul une valeur aléatoire entre pMin et pMax.*
- bool [util\\_sousTabExist](#) (Tour pChild, const int pBegin, const int pEnd, Tour pParent, bool pRecursvite)  
*Indique si un sous-tableau d'un tableau est présent dans un autre tableau.*

### 5.19.1 Description détaillée

Fonctions utiles.

#### Auteur

Antoine de Roquemaurel

#### Date

19/11/2012 16 :27 :39

Entêtes des fonctions pouvant être utiles dans tout le projet. Ce sont des fonctions simples, qui doivent être indépendantes du projet.

### 5.19.2 Documentation des fonctions

#### 5.19.2.1 void [util\\_displayArray](#) ( const int \* *pTab*, const int *pSize* )

Affiche le contenu d'un tableau d'entiers.

##### Paramètres

*pTab* Le tableau à afficher

*pSize* La taille du tableau

#### 5.19.2.2 int [util\\_rand](#) ( const int *pMin*, const int *pMax* )

Calcul une valeur aléatoire entre pMin et pMax.



**Paramètres**

*pMin* Minimum

*pMax* Maximum

**Renvoie****5.19.2.3 void util\_reverseArray ( Town \* pTab, const int pBegin, const int pEnd )**

Inverse les éléments d'un tableau pTab entre les cases pBegin et pEnd.

**Paramètres**

*pTab* Tableau à inverser

*pBegin* Début de la section à inverser

*pEnd* Fin de la section à inverser

**5.19.2.4 int util\_searchFirstOccurenceInArray ( char \*\* pArray, const int pSize, char \* pSearch )**

Cherche la première occurrence d'une chaîne de caractère dans un tableau de chaînes de caractères.

**Paramètres**

*pArray* Le tableau de chaînes de caractères dans lequel chercher

*pSize* La taille du tableau

*pSearch* La chaîne de caractère à chercher

**Renvoie**

La position de la chaîne dans le tableau ou -1 si elle n'a pas été trouvée

**5.19.2.5 bool util\_sousTabExist ( Tour pChild, const int pBegin, const int pEnd, Tour pParent, bool pRecursvite )**

Indique si un sous-tableau d'un tableau est présent dans un autre tableau.

**Paramètres**

*pChild* Le tableau à chercher

*pBegin* Le début de la section à chercher

*pEnd* La fin de la section à chercher

*pParent* Le tableau dans lequel chercher

*pRecursvite* Pour simplifier vis-à-vis de la récursivité, doit toujours être à false

**Renvoie**

Vrai si le sous-tableau a été trouvé faux sinon

**5.19.2.6 int util\_sum ( const int *pBegin*, const int *pEnd* )**

Calcul la somme des éléments allant de pBegin à pEnd.

**Paramètres**

*pBegin* Début de la somme

*pEnd* Fin de la somme

**Renvoie**

La somme des éléments

**5.19.2.7 void util\_swap ( int \* *a*, int \* *b* )**

Échange deux variables.

**Paramètres**

*a* Première variable à échanger

*b* Seconde variable

# Index

- Algo, [7](#)
- AlgoType
  - [parsing.h](#), [22](#)
- BRUTEFORCE
  - [parsing.h](#), [22](#)
- bruteForce.c
  - [bruteForce\\_bestPath](#), [30](#)
- bruteForce.h
  - [bruteForce\\_bestPath](#), [13](#)
- bruteForce\_bestPath
  - [bruteForce.c](#), [30](#)
  - [bruteForce.h](#), [13](#)
- Distance, [7](#)
- distance.c
  - [distance\\_betweenTowns](#), [31](#)
  - [distance\\_calculDistance](#), [31](#)
  - [distance\\_new](#), [31](#)
  - [distance\\_searchDistance](#), [31](#)
- distance.h
  - [distance\\_betweenTowns](#), [14](#)
  - [distance\\_calculDistance](#), [15](#)
  - [distance\\_new](#), [15](#)
  - [distance\\_searchDistance](#), [15](#)
- distance\_betweenTowns
  - [distance.c](#), [31](#)
  - [distance.h](#), [14](#)
- distance\_calculDistance
  - [distance.c](#), [31](#)
  - [distance.h](#), [15](#)
- distance\_new
  - [distance.c](#), [31](#)
  - [distance.h](#), [15](#)
- distance\_searchDistance
  - [distance.c](#), [31](#)
  - [distance.h](#), [15](#)
- END
  - [parsing.h](#), [22](#)
- Errors, [8](#)
- errors.h
  - [errors\\_displayErrorsMessage](#), [17](#)
  - [errors\\_new](#), [17](#)
  - [errors\\_setFileNotFound](#), [17](#)
  - [errors\\_setMissingParameterGa](#), [17](#)
  - [errors\\_setMissingParameterLsnr](#), [17](#)
  - [errors\\_setMissingParameterLsr](#), [17](#)
  - [errors\\_setNbArguments](#), [17](#)
  - [errors\\_setNoAlgoSpecified](#), [18](#)
  - [errors\\_setNoValidParameterLsnr](#), [18](#)
  - [errors\\_setNoValidParameterLsr](#), [18](#)
  - [errors\\_setTagFNotFound](#), [18](#)
- errors\_displayErrorsMessage
  - [errors.h](#), [17](#)
- errors\_new
  - [errors.h](#), [17](#)
- errors\_setFileNotFound
  - [errors.h](#), [17](#)
- errors\_setMissingParameterGa
  - [errors.h](#), [17](#)
- errors\_setMissingParameterLsnr
  - [errors.h](#), [17](#)
- errors\_setMissingParameterLsr
  - [errors.h](#), [17](#)
- errors\_setNbArguments
  - [errors.h](#), [17](#)
- errors\_setNoAlgoSpecified
  - [errors.h](#), [18](#)
- errors\_setNoValidParameterLsnr
  - [errors.h](#), [18](#)
- errors\_setNoValidParameterLsr
  - [errors.h](#), [18](#)
- errors\_setTagFNotFound
  - [errors.h](#), [18](#)
- GENETIC
  - [parsing.h](#), [22](#)
- Instance, [9](#)
- instance.c
  - [instance\\_display](#), [33](#)
  - [instance\\_displayLinearVector](#), [33](#)
  - [instance\\_displayMatrix](#), [33](#)
  - [instance\\_initializeDistancesMatrix](#), [33](#)
  - [instance\\_new](#), [34](#)
  - [instance\\_push](#), [34](#)
- instance.h
  - [instance\\_display](#), [19](#)
  - [instance\\_displayLinearVector](#), [19](#)

- instance\_displayMatrix, 20
- instance\_initializeDistancesMatrix, 20
- instance\_new, 20
- instance\_push, 20
- instance\_display
  - instance.c, 33
  - instance.h, 19
- instance\_displayLinearVector
  - instance.c, 33
  - instance.h, 19
- instance\_displayMatrix
  - instance.c, 33
  - instance.h, 20
- instance\_initializeDistancesMatrix
  - instance.c, 33
  - instance.h, 20
- instance\_new
  - instance.c, 34
  - instance.h, 20
- instance\_push
  - instance.c, 34
  - instance.h, 20
- lib/bruteForce.h, 13
- lib/distance.h, 14
- lib/errors.h, 15
- lib/instance.h, 18
- lib/localSearch.h, 21
- lib/parsing.h, 21
- lib/tour.h, 23
- lib/town.h, 25
- lib/util.h, 26
- LOCALSEARCH\_RANDOM
  - parsing.h, 22
- LOCALSEARCH\_SYSTEMATIC
  - parsing.h, 22
- main
  - main.c, 35
- main.c
  - main, 35
- parsing.c
  - parsing\_algoType, 36
  - parsing\_parseFileName, 36
  - parsing\_parseVerboseMode, 36
- parsing.h
  - AlgoType, 22
  - BRUTEFORCE, 22
  - END, 22
  - GENETIC, 22
  - LOCALSEARCH\_RANDOM, 22
  - LOCALSEARCH\_SYSTEMATIC, 22
  - parsing\_algoType, 22
  - parsing\_parseFileName, 23
  - parsing\_parseVerboseMode, 23
- parsing\_algoType
  - parsing.c, 36
  - parsing.h, 22
- parsing\_parseFileName
  - parsing.c, 36
  - parsing.h, 23
- parsing\_parseVerboseMode
  - parsing.c, 36
  - parsing.h, 23
- src/bruteForce.c, 29
- src/distance.c, 30
- src/genetic.c, 32
- src/instance.c, 32
- src/localSearch.c, 34
- src/main.c, 35
- src/parsing.c, 35
- src/tour.c, 37
- src/town.c, 39
- src/util.c, 39
- Tour, 10
- tour.c
  - tour\_2opt, 38
  - tour\_calculLength, 38
  - tour\_display, 38
  - tour\_new, 38
  - tour\_nextPermutation, 38
  - tour\_randomWalk, 38
- tour.h
  - tour\_2opt, 24
  - tour\_calculLength, 24
  - tour\_display, 25
  - tour\_new, 25
  - tour\_nextPermutation, 25
  - tour\_randomWalk, 25
- tour\_2opt
  - tour.c, 38
  - tour.h, 24
- tour\_calculLength
  - tour.c, 38
  - tour.h, 24
- tour\_display
  - tour.c, 38
  - tour.h, 25
- tour\_new
  - tour.c, 38
  - tour.h, 25
- tour\_nextPermutation
  - tour.c, 38
  - tour.h, 25
- tour\_randomWalk

- tour.c, [38](#)
  - tour.h, [25](#)
- Town, [10](#)
- town.c
  - town\_new, [39](#)
- town.h
  - town\_new, [26](#)
- town\_new
  - town.c, [39](#)
  - town.h, [26](#)
- util.c
  - util\_displayArray, [40](#)
  - util\_rand, [40](#)
  - util\_reverseArray, [41](#)
  - util\_searchFirstOccurenceInArray, [41](#)
  - util\_sousTabExist, [41](#)
  - util\_sum, [41](#)
  - util\_swap, [42](#)
- util.h
  - util\_displayArray, [27](#)
  - util\_rand, [27](#)
  - util\_reverseArray, [28](#)
  - util\_searchFirstOccurenceInArray, [28](#)
  - util\_sousTabExist, [28](#)
  - util\_sum, [28](#)
  - util\_swap, [29](#)
- util\_displayArray
  - util.c, [40](#)
  - util.h, [27](#)
- util\_rand
  - util.c, [40](#)
  - util.h, [27](#)
- util\_reverseArray
  - util.c, [41](#)
  - util.h, [28](#)
- util\_searchFirstOccurenceInArray
  - util.c, [41](#)
  - util.h, [28](#)
- util\_sousTabExist
  - util.c, [41](#)
  - util.h, [28](#)
- util\_sum
  - util.c, [41](#)
  - util.h, [28](#)
- util\_swap
  - util.c, [42](#)
  - util.h, [29](#)