

## Liste doublement chaînée

Généré par Doxygen 1.8.2

Samedi Décembre 1 2012 23 :16 :52



# Table des matières

<b>1</b>	<b>Index des structures de données</b>	<b>1</b>
1.1	Structures de données	1
<b>2</b>	<b>Index des fichiers</b>	<b>3</b>
2.1	Liste des fichiers	3
<b>3</b>	<b>Documentation des structures de données</b>	<b>5</b>
3.1	Référence de la structure Cellule_struct	5
3.1.1	Description détaillée	5
3.1.2	Documentation des champs	5
3.1.2.1	afficher	5
3.1.2.2	comparer	6
3.2	Référence de la structure Liste_struct	6
3.2.1	Description détaillée	7
3.2.2	Documentation des champs	7
3.2.2.1	a	7
3.2.2.2	afficher	7
3.2.2.3	dernierElement	7
3.2.2.4	egal	7
3.2.2.5	insererAIndex	8
3.2.2.6	insererDebut	8
3.2.2.7	insererFin	8
3.2.2.8	insererListeAIndex	8
3.2.2.9	insererListeDebut	8
3.2.2.10	insererListeFin	8
3.2.2.11	premierElement	8
3.2.2.12	sousListe	9
3.2.2.13	supprimerAIndex	9
3.2.2.14	supprimerDernier	9
3.2.2.15	supprimerPremier	9
3.2.2.16	trier	9
3.2.2.17	vider	9

3.3	Référence de la structure typeAssocie . . . . .	9
3.3.1	Description détaillée . . . . .	10
3.3.2	Documentation des champs . . . . .	10
3.3.2.1	fonctionAffichage . . . . .	10
3.3.2.2	fonctionCompare . . . . .	10
<b>4</b>	<b>Documentation des fichiers</b>	<b>11</b>
4.1	Référence du fichier liste.h . . . . .	11
4.1.1	Description détaillée . . . . .	13
4.1.2	Documentation des macros . . . . .	13
4.1.2.1	CELLULE_VALEUR . . . . .	13
4.1.3	Documentation des fonctions . . . . .	13
4.1.3.1	__CHAINE__affiche . . . . .	13
4.1.3.2	__CHAINE__compare . . . . .	13
4.1.3.3	__CHAR__affiche . . . . .	14
4.1.3.4	__CHAR__compare . . . . .	14
4.1.3.5	__DOUBLE__affiche . . . . .	14
4.1.3.6	__DOUBLE__compare . . . . .	14
4.1.3.7	__FLOAT__affiche . . . . .	14
4.1.3.8	__FLOAT__compare . . . . .	14
4.1.3.9	__INT__affiche . . . . .	15
4.1.3.10	__INT__compare . . . . .	15
4.1.3.11	__LONG__affiche . . . . .	15
4.1.3.12	__LONG__compare . . . . .	15
4.1.3.13	__SHORT__affiche . . . . .	15
4.1.3.14	__SHORT__compare . . . . .	16
4.1.3.15	Cellule_afficher . . . . .	16
4.1.3.16	Cellule_cloner . . . . .	16
4.1.3.17	Cellule_comparer . . . . .	16
4.1.3.18	Cellule_creer . . . . .	16
4.1.3.19	Cellule_detruire . . . . .	17
4.1.3.20	Cellule_detruireSiInutilise . . . . .	17
4.1.3.21	Cellule_egal . . . . .	17
4.1.3.22	Liste_a . . . . .	17
4.1.3.23	Liste_afficher . . . . .	17
4.1.3.24	Liste_cloner . . . . .	18
4.1.3.25	Liste_creer . . . . .	18
4.1.3.26	Liste_dernierElement . . . . .	18
4.1.3.27	Liste_detruire . . . . .	18
4.1.3.28	Liste_egal . . . . .	18

4.1.3.29 Liste_insererAIndex . . . . .	19
4.1.3.30 Liste_insererDebut . . . . .	19
4.1.3.31 Liste_insererFin . . . . .	19
4.1.3.32 Liste_insererListeAIndex . . . . .	19
4.1.3.33 Liste_insererListeDebut . . . . .	20
4.1.3.34 Liste_insererListeFin . . . . .	20
4.1.3.35 Liste_premierElement . . . . .	20
4.1.3.36 Liste_sousListe . . . . .	20
4.1.3.37 Liste_supprimerAIndex . . . . .	20
4.1.3.38 Liste_supprimerDernier . . . . .	21
4.1.3.39 Liste_supprimerPremier . . . . .	21
4.1.3.40 Liste_trier . . . . .	21
4.1.3.41 Liste_vider . . . . .	21
4.1.3.42 typeAssocie_creer . . . . .	22
4.1.3.43 typeAssocie_estEgal . . . . .	22
4.1.3.44 typeAssocie_estNull . . . . .	22

**Index****22**



# Chapitre 1

## Index des structures de données

### 1.1 Structures de données

Liste des structures de données avec une brève description :

<a href="#">Cellule_struct</a>	La structure Cellule est la representation d'un element d'une liste . . . . .	5
<a href="#">Liste_struct</a>	La structure Liste est une collection de cellule . . . . .	6
<a href="#">typeAssocie</a>	La structure permettant d'associer un type à une Liste ou une Cellule . . . . .	9





## Chapitre 2

# Index des fichiers

### 2.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

[liste.h](#)

Fichier .h des structures Cellule et Liste et leur fonctions . . . . . 11



# Chapitre 3

## Documentation des structures de données

### 3.1 Référence de la structure Cellule\_struct

la structure Cellule est la representation d'un element d'une liste.

```
#include <liste.h>
```

#### Champs de données

- void \* `data`  
*Contenu de la cellule.*
- struct `Cellule_struct` \* `previous`  
*Pointeur vers la cellule précédente.*
- struct `Cellule_struct` \* `next`  
*Pointeur vers la cellule suivante.*
- int `used`  
*Vaut 1 si cellule est presente dans une liste, sinon 0.*
- `typeAssocie ta`  
*Type associe à la cellule (par default int).*
- void(\* `afficher` )(struct `Cellule_struct` \*c)  
*Fonction d'affichage de la cellule.*
- int(\* `comparer` )(struct `Cellule_struct` \*c1, struct `Cellule_struct` \*c2)  
*Fonction de comparaison de 2 cellules.*

#### 3.1.1 Description détaillée

la structure Cellule est la representation d'un element d'une liste.

La Cellule est crée via l'appel de la fonction `Cellule_creer(void* data)`.

Un exemple d'utilisation simple pour creer une cellules contenant un int de valeur 5 :

```
int valeur = 5;
Cellule c = Cellule_creer(&valeur); // c contient la valeur 5, et peut
    être utilisé dans les listes
Cellule_detruire(c); // Ne pas oublier de deallouer la mémoire utilisée
```

#### 3.1.2 Documentation des champs

##### 3.1.2.1 void(\* afficher)(struct Cellule\_struct \*c)

Fonction d'affichage de la cellule.

Voir également

[\\_\\_CHAINE\\_\\_affiche\(\)](#), [\\_\\_INT\\_\\_affiche\(\)](#), [\\_\\_SHORT\\_\\_affiche\(\)](#), [\\_\\_LONG\\_\\_affiche\(\)](#), [\\_\\_CHAR\\_\\_affiche\(\)](#), [\\_\\_DOUBLE\\_\\_affiche\(\)](#), [\\_\\_FLOAT\\_\\_affiche\(\)](#).

### 3.1.2.2 `int(*comparer)(struct Cellule_struct *c1, struct Cellule_struct *c2)`

Fonction de comparaison de 2 cellules.

Voir également

[\\_\\_CHAINE\\_\\_compare\(\)](#), [\\_\\_INT\\_\\_compare\(\)](#), [\\_\\_SHORT\\_\\_compare\(\)](#), [\\_\\_LONG\\_\\_compare\(\)](#), [\\_\\_CHAR\\_\\_affiche\(\)](#), [\\_\\_DOUBLE\\_\\_affiche\(\)](#), [\\_\\_FLOAT\\_\\_affiche\(\)](#).

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

– [liste.h](#)

## 3.2 Référence de la structure `Liste_struct`

la structure Liste est une collection de cellule.

```
#include <liste.h>
```

### Champs de données

- [Cellule entry](#)  
*Point d'entrée dans la suite des Cellules.*
- `int size`  
*Mémoire la taille ( nombre de cellules) de la liste.*
- `Cellule(* a)(struct Liste_struct *l, int index)`  
*Renvoie l'element d'index correspondant.*
- `Cellule(* premierElement)(struct Liste_struct *l)`  
*Renvoie le premier element.*
- `Cellule(* dernierElement)(struct Liste_struct *l)`  
*Renvoie le dernier element.*
- `int(* egal)(struct Liste_struct *l1, struct Liste_struct *l2)`  
*Renvoie 1 si les deux listes sont égales sinon 0.*
- `int(* insererDebut)(struct Liste_struct *l, Cellule c)`  
*Insère une cellule au début de la liste.*
- `int(* insererFin)(struct Liste_struct *l, Cellule c)`  
*Insère une cellule à la fin de la liste.*
- `int(* insererAIndex)(struct Liste_struct *l, Cellule c, int index)`  
*Insère une cellule à l'index fourni tel que l.insererAIndex(l,c,1) -> l.a(l,1) == c.*
- `int(* supprimerPremier)(struct Liste_struct *l)`  
*Supprime la première cellule de la liste.*
- `int(* supprimerDernier)(struct Liste_struct *l)`  
*Supprime la dernière cellule de la liste.*
- `int(* supprimerAIndex)(struct Liste_struct *l, int index)`  
*Supprime la cellule à l'index donné.*
- `int(* vider)(struct Liste_struct *l)`  
*Vide la liste (Supprime tout les éléments).*
- `int(* insererListeDebut)(struct Liste_struct *l, struct Liste_struct *lInserer)`  
*Insère la deuxième liste dans la première au début (recopie de la deuxième dans la première).*
- `int(* insererListeFin)(struct Liste_struct *l, struct Liste_struct *lInserer)`  
*Insère la deuxième liste dans la première à la fin (recopie de la deuxième dans la première).*
- `int(* insererListeAIndex)(struct Liste_struct *l, struct Liste_struct *lInserer, int index)`  
*Insère la deuxième liste dans la première à l'index (recopie de la deuxième dans la première).*
- `struct Liste_struct *(* sousListe)(struct Liste_struct *l, int index)`  
*Renvoie la sous-liste, extrai à partir de l'index jusqu'à la fin de la liste.*
- `typeAssocie ta`  
*Type Associe à la liste.*
- `void(* afficher)(struct Liste_struct *l)`

- Affiche la liste grâce au fonction "afficher" de ta, si ta est null, il y'a appelle à afficher de la Cellule.*
- `int(*trier)(struct Liste_struct *l)`  
*Trie une liste dans l'ordre croissant, il faut que le type associé soit non-null.*

### 3.2.1 Description détaillée

la structure Liste est une collection de cellule.

La liste est une structure permettant de représenter une liste doublement chaînée.

Un certain nombre de fonction ont été mise en oeuvre pour vous garantir une facilité d'utilisation :

- Constructeur et destructeur.
- Recupération de cellule.
- Insertion de cellule/liste.
- Creation de sous-liste.
- Triage.

Le code basique pour la creation d'une liste d'entier (int) sera :

```
int valeur = 5;
Liste l = Liste_creer(NULL,associe_int); // Creer une liste d'entier vide
Liste_insérerDebut(l, Cellule_creer(&valeur); // Insertion de 5 dans la liste
Liste_detruire(l); // Libere la mémoire de l et de tout les elements qu'elle
comporte
```

### 3.2.2 Documentation des champs

#### 3.2.2.1 Cellule(\* a)(struct Liste\_struct \*l, int index)

Renvoie l'element d'index correspondant.

Voir également

[Liste\\_a\(\)](#)

#### 3.2.2.2 void(\* afficher)(struct Liste\_struct \*l)

Affiche la liste grâce au fonction "afficher" de ta, si ta est null, il y'a appelle à afficher de la Cellule.

Voir également

[Liste\\_afficher\(\)](#)

#### 3.2.2.3 Cellule(\* dernierElement)(struct Liste\_struct \*l)

Renvoie le dernier element.

Voir également

[Liste\\_dernierElement\(\)](#)

#### 3.2.2.4 int(\* egal)(struct Liste\_struct \*l1, struct Liste\_struct \*l2)

Renvoie 1 si les deux listes sont égales sinon 0.

Voir également

[Liste\\_egal\(\)](#)

**3.2.2.5 int(\* insererAIndex)(struct Liste\_struct \*l, Cellule c, int index)**

Insère une cellule à l'index fourni tel que  $l.insererAIndex(l,c,1) \rightarrow l.a(1) == c$ .

Voir également

[Liste\\_insererAIndex\(\)](#)

**3.2.2.6 int(\* insererDebut)(struct Liste\_struct \*l, Cellule c)**

Insère une cellule au début de la liste.

Voir également

[Liste\\_insererDebut\(\)](#)

**3.2.2.7 int(\* insererFin)(struct Liste\_struct \*l, Cellule c)**

Insère une cellule à la fin de la liste.

Voir également

[Liste\\_insererFin\(\)](#)

**3.2.2.8 int(\* insererListeAIndex)(struct Liste\_struct \*l, struct Liste\_struct \*lInserer, int index)**

Insère la deuxième liste dans la première à l'index (recopie de la deuxième dans la première).

Voir également

[Liste\\_insererListeAIndex\(\)](#)

**3.2.2.9 int(\* insererListeDebut)(struct Liste\_struct \*l, struct Liste\_struct \*lInserer)**

Insère la deuxième liste dans la première au début (recopie de la deuxième dans la première).

Voir également

[Liste\\_insererListeDebut\(\)](#)

**3.2.2.10 int(\* insererListeFin)(struct Liste\_struct \*l, struct Liste\_struct \*lInserer)**

Insère la deuxième liste dans la première à la fin (recopie de la deuxième dans la première).

Voir également

[Liste\\_insererListeFin\(\)](#)

**3.2.2.11 Cellule(\* premierElement)(struct Liste\_struct \*l)**

Renvoie le premier element.

Voir également

[Liste\\_premierElement\(\)](#)

**3.2.2.12 struct Liste\_struct\*(\* sousListe)(struct Liste\_struct \*l, int index) [read]**

Renvoie la sous-liste, extrait à partir de l'index jusqu'à la fin de la liste.

Voir également

[Liste\\_sousListe\(\)](#)

**3.2.2.13 int(\* supprimerAIndex)(struct Liste\_struct \*l, int index)**

Supprime la cellule à l'index donné.

Voir également

[Liste\\_supprimerAIndex\(\)](#)

**3.2.2.14 int(\* supprimerDernier)(struct Liste\_struct \*l)**

Supprime la dernière cellule de la liste.

Voir également

[Liste\\_supprimerDernier](#)

**3.2.2.15 int(\* supprimerPremier)(struct Liste\_struct \*l)**

Supprime la première cellule de la liste.

Voir également

[Liste\\_supprimerPremier\(\)](#)

**3.2.2.16 int(\* trier)(struct Liste\_struct \*l)**

Trie une liste dans l'ordre croissant, il faut que le type associé soit non-null.

Voir également

[Liste\\_trier\(\)](#)

**3.2.2.17 int(\* vider)(struct Liste\_struct \*l)**

Vide la liste (Supprime tout les éléments).

Voir également

[Liste\\_vider\(\)](#)

La documentation de cette structure a été générée à partir du fichier suivant :  
– [liste.h](#)

## 3.3 Référence de la structure typeAssocie

la structure permettant d'associer un type à une Liste ou une Cellule.

```
#include <liste.h>
```

## Champs de données

- void(\* [fonctionAffichage](#) )(void \*data)  
*Affiche sur la sortie standard data.*
- int(\* [fonctionCompare](#) )(const void \*data1, const void \*data2)  
*Compare data1 et data2, renvoie -1 si data1 < data2, 0 si data1 = data2, 1 si data1 > data2.*
- int(\* [estNull](#) )(struct [typeAssocie](#) ta)  
*Renvoie 1 si la fonction d'affichage et la fonction de comparaison ne sont pas définies, sinon 0.*
- int(\* [estEgal](#) )(struct [typeAssocie](#) ta1, struct [typeAssocie](#) ta2)  
*Renvoie 1 si la structure ta1 et ta2 sont identiques, sinon 0.*

### 3.3.1 Description détaillée

la structure permettant d'associer un type à une Liste ou une Cellule.

[typeAssocie](#) est une simple interface permettant quelques operations :

- L'affichage d'une liste ou d'une cellule sur la sortie standard (par default).
- La comparaison de deux cellules.
- Le triage d'une liste.

Les types primitifs sont déjà implementés voir [associe\\_chaine](#), [associe\\_int](#), [associe\\_short](#), [associe\\_long](#), [associe\\_float](#), [associe\\_double](#), [associe\\_char](#), [associe\\_void](#).

Si votre liste est heterogene, veuillez utiliser le type : [associe\\_void](#).

Si il vous faut utiliser des types personnalisés pour vos listes ou vos cellules, le code basique sera :

```
typeAssocie monTa = typeAssocie_creer (&maFctAffichage, &
    maFctDeComparaison);
Liste l = Liste_creer (NULL, monTa);
```

### 3.3.2 Documentation des champs

#### 3.3.2.1 void(\* fonctionAffichage)(void \*data)

Affiche sur la sortie standard data.

Voir également

[\\_\\_CHaine\\_\\_affiche\(\)](#), [\\_\\_INT\\_\\_affiche\(\)](#), [\\_\\_SHORT\\_\\_affiche\(\)](#), [\\_\\_LONG\\_\\_affiche\(\)](#), [\\_\\_CHAR\\_\\_affiche\(\)](#), [\\_\\_DOUBLE\\_\\_affiche\(\)](#), [\\_\\_FLOAT\\_\\_affiche\(\)](#).

#### 3.3.2.2 int(\* fonctionCompare)(const void \*data1, const void \*data2)

Compare data1 et data2, renvoie -1 si data1 < data2, 0 si data1 = data2, 1 si data1 > data2.

Voir également

[\\_\\_CHaine\\_\\_compare\(\)](#), [\\_\\_INT\\_\\_compare\(\)](#), [\\_\\_SHORT\\_\\_compare\(\)](#), [\\_\\_LONG\\_\\_compare\(\)](#), [\\_\\_CHAR\\_\\_affiche\(\)](#), [\\_\\_DOUBLE\\_\\_affiche\(\)](#), [\\_\\_FLOAT\\_\\_affiche\(\)](#).

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

- [liste.h](#)



# Chapitre 4

## Documentation des fichiers

### 4.1 Référence du fichier liste.h

fichier .h des structures Cellule et Liste et leur fonctions

```
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <string.h>
```

#### Structures de données

- struct `typeAssocie`  
*la structure permettant d'associer un type à une Liste ou une Cellule.*
- struct `Cellule_struct`  
*la structure Cellule est la representation d'un element d'une liste.*
- struct `Liste_struct`  
*la structure Liste est une collection de cellule.*

#### Macros

- #define `CELLULE_VALEUR(Cellule)`

#### Définitions de type

- typedef struct `typeAssocie` `typeAssocie`  
*"struct typeAssocie" simplifié en "typeAssocie".*
- typedef struct `Cellule_struct` \* `Cellule`  
*pointeur vers la structure Cellule.*
- typedef `Liste_struct` \* `Liste`  
*pointeur vers la structure Liste.*

#### Fonctions

- int `Cellule_egal` (`Cellule` c1, `Cellule` c2)  
*Teste l'égalité de deux cellules.*
- int `Liste_egal` (`Liste` l1, `Liste` l2)  
*Teste l'égalité deux listes.*
- `Cellule` `Cellule_cloner` (`Cellule` c)  
*Copie une cellule.*
- `Liste` `Liste_cloner` (`Liste` l)  
*Copie une liste.*

- int `Cellule_comparer` (`Cellule` c1, `Cellule` c2)  
*Compare 2 cellules.*
- `typeAssocie` `typeAssocie_creer` (void fctAffichage(void \*), int fctCompare(const void \*, const void \*))  
*Crée un type associé.*
- `Cellule` `Cellule_creer` (void \*data)  
*Crée une cellule.*
- `Liste` `Liste_creer` (`Cellule` c, `typeAssocie` ta)  
*Crée une liste à partir d'une cellule.*
- int `Cellule_detruire` (`Cellule` c)  
*Détruit une cellule.*
- int `Cellule_detruireSiInutilise` (`Cellule` c)  
*Détruit une cellule inutilisée.*
- int `Liste_detruire` (`Liste` l)  
*Détruit une liste.*
- `Cellule` `Liste_a` (`Liste` l, int index)  
*Accès à une cellule avec une certaine position dans une liste.*
- `Cellule` `Liste_premierElement` (`Liste` l)  
*Accès au 1er élément de la liste.*
- `Cellule` `Liste_dernierElement` (`Liste` l)  
*Accès au dernier élément de la liste.*
- int `Liste_insererDebut` (`Liste` l, `Cellule` c)  
*Insere une cellule au début d'une liste.*
- int `Liste_insererFin` (`Liste` l, `Cellule` c)  
*Insere une cellule à la fin d'une liste.*
- int `Liste_insererAIndex` (`Liste` l, `Cellule` c, int index)  
*Insere une cellule dans une certaine place d'une liste.*
- int `Liste_supprimerPremier` (`Liste` l)  
*Supprime le 1er élément d'une liste.*
- int `Liste_supprimerDernier` (`Liste` l)  
*Supprime le dernier élément d'une liste.*
- int `Liste_supprimerAIndex` (`Liste` l, int index)  
*Supprime la cellule a l'index indique dans une liste.*
- int `Liste_vider` (`Liste` l)  
*Vide une liste.*
- int `Liste_insererListeDebut` (`Liste` l, `Liste` lInserer)  
*Insere une liste au début d'une autre liste.*
- int `Liste_insererListeFin` (`Liste` l, `Liste` lInserer)  
*Insere une liste à la fin d'une autre liste.*
- int `Liste_insererListeAIndex` (`Liste` l, `Liste` lInserer, int index)  
*Insere une liste dans d'une autre liste à un index spécifié.*
- `Liste` `Liste_sousListe` (`Liste` l, int index)  
*Renvoie la sous-liste d'une liste à partir d'une position.*
- void `Liste_afficher` (`Liste` l)  
*Affiche le contenu d'une liste.*
- int `Liste_trier` (`Liste` l)  
*Trie une liste.*
- void `Cellule_afficher` (`Cellule` c)  
*Affiche le contenu d'une cellule.*
- int `typeAssocie_estNull` (`typeAssocie` ta)  
*Teste si un type associé est null.*
- int `typeAssocie_estEgal` (`typeAssocie` ta1, `typeAssocie` ta2)  
*Teste l'égalité de 2 types associés.*
- void `__CHAINE__affiche` (void \*ch)  
*Affiche le contenu d'une chaine de caractères.*
- int `__CHAINE__compare` (const void \*ch1, const void \*ch2)  
*Comparer 2 chaines de caractères.*
- void `__INT__affiche` (void \*in)  
*Affiche le contenu d'un int.*
- int `__INT__compare` (const void \*in1, const void \*in2)  
*Compare 2 int.*
- void `__SHORT__affiche` (void \*sh)  
*Affiche le contenu d'un short.*
- int `__SHORT__compare` (const void \*sh1, const void \*sh2)  
*Comparer 2 short.*
- void `__LONG__affiche` (void \*lo)  
*Affiche le contenu d'un long.*
- int `__LONG__compare` (const void \*lo1, const void \*lo2)  
*Comparer 2 long.*
- void `__FLOAT__affiche` (void \*fl)  
*Affiche le contenu d'un float.*

- int `__FLOAT__compare` (const void \*fl1, const void \*fl2)  
*Compare 2 float.*
- void `__DOUBLE__affiche` (void \*dou)  
*Affiche le contenu d'un double.*
- int `__DOUBLE__compare` (const void \*dou1, const void \*dou2)  
*Compare 2 double.*
- void `__CHAR__affiche` (void \*ch)  
*Afficher le contenu d'un char.*
- int `__CHAR__compare` (const void \*ch1, const void \*ch2)  
*Comparer 2 char.*

### 4.1.1 Description détaillée

fichier .h des structures Cellule et Liste et leur fonctions

### 4.1.2 Documentation des macros

#### 4.1.2.1 #define CELLULE\_VALEUR( Cellule )

Valeur :

```
!memcmp (&(Cellule->ta), &(associe_int), sizeof(typeAssocie))
    ? *(int*)Cellule->data : \
!memcmp (&(Cellule->ta), &(associe_short), sizeof(typeAssocie
)) ? *(short*)Cellule->data : \
!memcmp (&(Cellule->ta), &(associe_long), sizeof(typeAssocie
)) ? *(long*)Cellule->data : \
!memcmp (&(Cellule->ta), &(associe_float), sizeof(typeAssocie
)) ? *(float*)Cellule->data : \
!memcmp (&(Cellule->ta), &(associe_double), sizeof(typeAssocie
)) ? *(double*)Cellule->data : \
!memcmp (&(Cellule->ta), &(associe_char), sizeof(typeAssocie
)) ? *(char*)Cellule->data : 0\
```

Renvoie la valeur d'une cellule selon son type associe.

### 4.1.3 Documentation des fonctions

#### 4.1.3.1 void \_\_CHAINE\_\_affiche ( void \* ch )

Affiche le contenu d'une chaine de caractères.

Paramètres

<i>ch</i>	la chaine à afficher.
-----------	-----------------------

#### 4.1.3.2 int \_\_CHAINE\_\_compare ( const void \* ch1, const void \* ch2 )

Comparer 2 chaines de caractères.

Paramètres

<i>ch1, ch2</i>	2 chaines à comparer.
-----------------	-----------------------

**Renvoie**

0 si elles sont égales, 1 si  $ch1 > ch2$ , -1 si  $ch1 < ch2$ .

**4.1.3.3 void \_\_CHAR\_\_affiche ( void \* *ch* )**

Afficher le contenu d'un char.

**Paramètres**

<i>ch</i>	le char à afficher.
-----------	---------------------

**4.1.3.4 int \_\_CHAR\_\_compare ( const void \* *ch1*, const void \* *ch2* )**

Comparer 2 char.

**Paramètres**

<i>ch1,ch2</i>	2 double à comparer.
----------------	----------------------

**Renvoie**

0 si  $ch1 == ch2$ , 1 si  $ch1 > ch2$ , -1 si  $ch1 < ch2$ .

**4.1.3.5 void \_\_DOUBLE\_\_affiche ( void \* *dou* )**

Affiche le contenu d'un double.

**Paramètres**

<i>dou</i>	le double à afficher.
------------	-----------------------

**4.1.3.6 int \_\_DOUBLE\_\_compare ( const void \* *dou1*, const void \* *dou2* )**

Compare 2 double.

**Paramètres**

<i>dou1,dou2</i>	2 double à comparer.
------------------	----------------------

**Renvoie**

0 si  $dou1 == dou2$ , 1 si  $dou1 > dou2$ , -1 si  $dou1 < dou2$ .

**4.1.3.7 void \_\_FLOAT\_\_affiche ( void \* *fl* )**

Affiche le contenu d'un float.

**Paramètres**

<i>fl</i>	le float à afficher.
-----------	----------------------

**4.1.3.8 int \_\_FLOAT\_\_compare ( const void \* *fl1*, const void \* *fl2* )**

Compare 2 float.

## Paramètres

<i>fl1,fl2</i>	2 float à comparer.
----------------	---------------------

## Renvoie

0 si  $fl1 == fl2$ , 1 si  $fl1 > fl2$ , -1 si  $fl1 < fl2$ .

**4.1.3.9 void \_\_INT\_\_affiche ( void \* *in* )**

Affiche le contenu d'un int.

## Paramètres

<i>in</i>	le int à afficher.
-----------	--------------------

**4.1.3.10 int \_\_INT\_\_compare ( const void \* *in1*, const void \* *in2* )**

Compare 2 int.

## Paramètres

<i>in1,in2</i>	2 int à comparer.
----------------	-------------------

## Renvoie

0 si  $in1 == in2$ , 1 si  $in1 > in2$ , -1 si  $in1 < in2$ .

**4.1.3.11 void \_\_LONG\_\_affiche ( void \* *lo* )**

Affiche le contenu d'un long.

## Paramètres

<i>lo</i>	le long à afficher.
-----------	---------------------

**4.1.3.12 int \_\_LONG\_\_compare ( const void \* *lo1*, const void \* *lo2* )**

Comparer 2 long.

## Paramètres

<i>lo1,lo2</i>	2 long à comparer.
----------------	--------------------

## Renvoie

0 si  $lo1 == lo2$ , 1 si  $lo1 > lo2$ , -1 si  $lo1 < lo2$ .

**4.1.3.13 void \_\_SHORT\_\_affiche ( void \* *sh* )**

Affiche le contenu d'un short.

## Paramètres

<i>sh</i>	le short à afficher
-----------	---------------------

**4.1.3.14 int \_\_SHORT\_\_compare ( const void \* *sh1*, const void \* *sh2* )**

Comparer 2 short.

**Paramètres**

<i>sh1,sh2</i>	2 in à comparer.
----------------	------------------

**Renvoie**

0 si  $sh1 == sh2$ , 1 si  $sh1 > sh2$ , -1 si  $sh1 < sh2$ .

**4.1.3.15 void Cellule\_afficher ( Cellule *c* )**

Affiche le contenu d'une cellule.

**Paramètres**

<i>c</i>	la cellule à afficher.
----------	------------------------

**4.1.3.16 Cellule Cellule\_cloner ( Cellule *c* )**

Copie une cellule.

**Renvoie**

le pointeur vers la copie de la cellule.

**Paramètres**

<i>c</i>	la cellule à copier.
----------	----------------------

**4.1.3.17 int Cellule\_comparer ( Cellule *c1*, Cellule *c2* )**

Compare 2 cellules.

**Paramètres**

<i>c1,c2</i>	2 cellules a comparer.
--------------	------------------------

**Renvoie**

-2 si *c1* et *c2* sont de types differents, -1 si  $c1 < c2$ , 0 si  $c1 = c2$ , 1 si  $c1 > c2$ .

**4.1.3.18 Cellule Cellule\_creer ( void \* *data* )**

Crée une cellule.

**Paramètres**

<i>data</i>	l'information a insérer dans la cellule.
-------------	--

**Renvoie**

le pointeur vers la cellule créée.

**4.1.3.19 int Cellule\_detruire ( Cellule c )**

Détruit une cellule.

**Paramètres**

c	la cellule à détruire.
---	------------------------

**Renvoie**

-1 si erreur, 0 si la cellule est détruite.

**4.1.3.20 int Cellule\_detruireSiInutilise ( Cellule c )**

Détruit une cellule inutilisée.

**Paramètres**

c	la cellule à détruire.
---	------------------------

La cellule n'est détruite uniquement si elle n'est présente dans aucune liste.

**Renvoie**

-1 si erreur, 0 si la cellule est détruite.

**4.1.3.21 int Cellule\_egal ( Cellule c1, Cellule c2 )**

Teste l'égalité de deux cellules.

**Paramètres**

c1,c2	deux cellules à tester
-------	------------------------

**Renvoie**

1 si deux cellules sont égaux, sinon 0.

**4.1.3.22 Cellule Liste\_a ( Liste l, int index )**

Accès à une cellule avec une certaine position dans une liste.

**Paramètres**

l	la liste.
index	position de la cellule dans la liste.

**Renvoie**

Le pointeur vers la cellule.

**4.1.3.23 void Liste\_afficher ( Liste l )**

Affiche le contenu d'une liste.

## Paramètres

/	la liste à afficher.
---	----------------------

**4.1.3.24 Liste Liste\_cloner ( Liste / )**

Copie une liste.

## Renvoie

le pointeur vers la copie de la liste.

## Paramètres

/	la liste à copier.
---	--------------------

**4.1.3.25 Liste Liste\_creer ( Cellule c, typeAssocie ta )**

Crée une liste à partir d'une cellule.

## Paramètres

c	la cellule que contient la liste.
ta	le <a href="#">typeAssocie</a> à la liste.

## Renvoie

le pointeur vers la liste créée.

**4.1.3.26 Cellule Liste\_dernierElement ( Liste / )**

Accès au dernier élément de la liste.

## Paramètres

/	la liste.
---	-----------

## Renvoie

Pointeur vers la dernière cellule.

**4.1.3.27 int Liste\_detruire ( Liste / )**

Détruit une liste.

## Paramètres

/	la liste à détruire.
---	----------------------

## Renvoie

-1 si erreur, 0 si la liste est détruite.

**4.1.3.28 int Liste\_egal ( Liste /1, Liste /2 )**

Teste l'égalité deux listes.



## Paramètres

<i>l1,l2</i>	deux listes à tester.
--------------	-----------------------

## Renvoie

1 si deux listes sont égaux, sinon 0.

**4.1.3.29 int Liste\_insererAtIndex ( Liste *l*, Cellule *c*, int *index* )**

Insere une cellule dans une certaine place d'une liste.

## Paramètres

<i>l</i>	la liste.
<i>c</i>	la cellule à insérer dans la liste.
<i>index</i>	la place pour insérer.

## Renvoie

-1 si erreur, 0 si succès.

**4.1.3.30 int Liste\_insererDebut ( Liste *l*, Cellule *c* )**

Insere une cellule au début d'une liste.

## Paramètres

<i>l</i>	la liste.
<i>c</i>	la cellule a insérer dans la liste.

## Renvoie

-1 si erreur, 0 si succès.

**4.1.3.31 int Liste\_insererFin ( Liste *l*, Cellule *c* )**

Insere une cellule à la fin d'une liste.

## Paramètres

<i>l</i>	la liste.
<i>c</i>	la cellule à insérer dans la liste.

## Renvoie

-1 si erreur, 0 si succès.

**4.1.3.32 int Liste\_insererListeAtIndex ( Liste *l*, Liste *lAInsérer*, int *index* )**

Insere une liste dans d'une autre liste à un index spécifié.

## Paramètres

<i>l</i>	la liste où on insère.
<i>lAInsérer</i>	la liste à insérer.
<i>index</i>	la position à inserer.

**Renvoie**

-1 si erreur : au moins 1 liste nulle ou trop courte ( index > taille ), 0 si succès.

**4.1.3.33 int Liste\_insererListeDebut ( Liste *l*, Liste *lAInserer* )**

Insere une liste au début d'une autre liste.

**Paramètres**

<i>l</i>	la liste où on insère.
<i>lAInserer</i>	la liste à insérer.

**Renvoie**

-1 si erreur : au moins 1 liste null, 0 si succès.

**4.1.3.34 int Liste\_insererListeFin ( Liste *l*, Liste *lAInserer* )**

Insere une liste à la fin d'une autre liste.

**Paramètres**

<i>l</i>	la liste où on insère.
<i>lAInserer</i>	la liste à insérer.

**Renvoie**

-1 si erreur : au moins 1 liste nulle, 0 si succès.

**4.1.3.35 Cellule Liste\_premierElement ( Liste *l* )**

Accès au 1er élément de la liste.

**Paramètres**

<i>l</i>	la liste.
----------	-----------

**Renvoie**

Pointeur vers la 1ere cellule.

**4.1.3.36 Liste Liste\_sousListe ( Liste *l*, int *index* )**

Renvoie la sous-liste d'une liste à partir d'une position.

**Paramètres**

<i>l</i>	la liste.
<i>index</i>	la position.

**Renvoie**

Pointeur vers la sous-liste.

**4.1.3.37 int Liste\_supprimerAIndex ( Liste *l*, int *index* )**

Supprime la cellule a l'index indique dans une liste.

## Paramètres

/	la liste.
index	l'index où on supprime la cellule.

## Renvoie

0 si succes, -1 si liste nulle, -2 si liste trop courte (taille<index).

**4.1.3.38 int Liste\_supprimerDernier ( Liste / )**

Supprime le dernier élément d'une liste.

## Paramètres

/	la liste.
---	-----------

## Renvoie

0 si succes, -1 si liste nulle, -2 si liste trop courte (taille 0).

**4.1.3.39 int Liste\_supprimerPremier ( Liste / )**

Supprime le 1er élément d'une liste.

## Paramètres

/	la liste.
---	-----------

## Renvoie

0 si succes, -1 si liste nulle, -2 si liste trop courte (taille 0).

**4.1.3.40 int Liste\_trier ( Liste / )**

Trie une liste.

## Paramètres

/	la liste à trier.
---	-------------------

## Renvoie

-1 si liste null ou sans définition de fonction de comparaison, 0 si succès.

**4.1.3.41 int Liste\_vider ( Liste / )**

Vide une liste.

## Paramètres

/	la liste à vider.
---	-------------------

**Renvoie**

0 si succès, -1 si la liste est null.

**4.1.3.42 typeAssocie typeAssocie\_creer ( void *fctAffichage*void \*, int *fctCompare*const void \*, const void \* )**

Crée un type associé.

**Paramètres**

<i>fctAffichage</i>	(void *) la fonction affichage propre du type associé.
<i>fctCompare</i>	(void *, void *) la fonction comparaison propre du type associé.

**Renvoie**

le type associe crée.

**4.1.3.43 int typeAssocie\_estEgal ( typeAssocie *ta1*, typeAssocie *ta2* )**

Teste l'égalité de 2 types associes.

**Paramètres**

<i>ta1,ta2</i>	2 types associés à comparer.
----------------	------------------------------

**Renvoie**

1 s'ils sont egaux, sinon 0.

**4.1.3.44 int typeAssocie\_estNull ( typeAssocie *ta* )**

Teste si un type associé est null.

**Paramètres**

<i>ta</i>	type associé à tester.
-----------	------------------------

**Renvoie**

1 si nul, sinon 0.

# Index

\_\_CHAINE\_\_affiche  
    liste.h, [13](#)  
\_\_CHAINE\_\_compare  
    liste.h, [13](#)  
\_\_CHAR\_\_affiche  
    liste.h, [14](#)  
\_\_CHAR\_\_compare  
    liste.h, [14](#)  
\_\_DOUBLE\_\_affiche  
    liste.h, [14](#)  
\_\_DOUBLE\_\_compare  
    liste.h, [14](#)  
\_\_FLOAT\_\_affiche  
    liste.h, [14](#)  
\_\_FLOAT\_\_compare  
    liste.h, [14](#)  
\_\_INT\_\_affiche  
    liste.h, [15](#)  
\_\_INT\_\_compare  
    liste.h, [15](#)  
\_\_LONG\_\_affiche  
    liste.h, [15](#)  
\_\_LONG\_\_compare  
    liste.h, [15](#)  
\_\_SHORT\_\_affiche  
    liste.h, [15](#)  
\_\_SHORT\_\_compare  
    liste.h, [15](#)

a

    Liste\_struct, [7](#)

afficher

    Cellule\_struct, [5](#)

    Liste\_struct, [7](#)

CELLULE\_VALEUR

    liste.h, [13](#)

Cellule\_afficher

    liste.h, [16](#)

Cellule\_cloner

    liste.h, [16](#)

Cellule\_comparer

    liste.h, [16](#)

Cellule\_creer

    liste.h, [16](#)

Cellule\_detruire

    liste.h, [17](#)

Cellule\_detruireSilnutilise

    liste.h, [17](#)

Cellule\_egal

    liste.h, [17](#)

Cellule\_struct, [5](#)

    afficher, [5](#)

    comparer, [6](#)

comparer

    Cellule\_struct, [6](#)

dernierElement

    Liste\_struct, [7](#)

egal

    Liste\_struct, [7](#)

fonctionAffichage

    typeAssocie, [10](#)

fonctionCompare

    typeAssocie, [10](#)

insérerAIndex

    Liste\_struct, [7](#)

insérerDebut

    Liste\_struct, [8](#)

insérerFin

    Liste\_struct, [8](#)

insérerListeAIndex

    Liste\_struct, [8](#)

insérerListeDebut

    Liste\_struct, [8](#)

insérerListeFin

    Liste\_struct, [8](#)

liste.h, [11](#)

    \_\_CHAINE\_\_affiche, [13](#)

    \_\_CHAINE\_\_compare, [13](#)

    \_\_CHAR\_\_affiche, [14](#)

    \_\_CHAR\_\_compare, [14](#)

    \_\_DOUBLE\_\_affiche, [14](#)

    \_\_DOUBLE\_\_compare, [14](#)

    \_\_FLOAT\_\_affiche, [14](#)

    \_\_FLOAT\_\_compare, [14](#)

    \_\_INT\_\_affiche, [15](#)

    \_\_INT\_\_compare, [15](#)

    \_\_LONG\_\_affiche, [15](#)

    \_\_LONG\_\_compare, [15](#)

    \_\_SHORT\_\_affiche, [15](#)

    \_\_SHORT\_\_compare, [15](#)

    CELLULE\_VALEUR, [13](#)

    Cellule\_afficher, [16](#)

    Cellule\_cloner, [16](#)

    Cellule\_comparer, [16](#)

    Cellule\_creer, [16](#)

Cellule\_detruire, [17](#)  
 Cellule\_detruireSiInutilise, [17](#)  
 Cellule\_egal, [17](#)  
 Liste\_a, [17](#)  
 Liste\_afficher, [17](#)  
 Liste\_cloner, [18](#)  
 Liste\_creer, [18](#)  
 Liste\_dernierElement, [18](#)  
 Liste\_detruire, [18](#)  
 Liste\_egal, [18](#)  
 Liste\_insererAIndex, [19](#)  
 Liste\_insererDebut, [19](#)  
 Liste\_insererFin, [19](#)  
 Liste\_insererListeAIndex, [19](#)  
 Liste\_insererListeDebut, [20](#)  
 Liste\_insererListeFin, [20](#)  
 Liste\_premierElement, [20](#)  
 Liste\_sousListe, [20](#)  
 Liste\_supprimerAIndex, [20](#)  
 Liste\_supprimerDernier, [21](#)  
 Liste\_supprimerPremier, [21](#)  
 Liste\_trier, [21](#)  
 Liste\_vider, [21](#)  
 typeAssocie\_creer, [22](#)  
 typeAssocie\_estEgal, [22](#)  
 typeAssocie\_estNull, [22](#)  
 Liste\_a  
     liste.h, [17](#)  
 Liste\_afficher  
     liste.h, [17](#)  
 Liste\_cloner  
     liste.h, [18](#)  
 Liste\_creer  
     liste.h, [18](#)  
 Liste\_dernierElement  
     liste.h, [18](#)  
 Liste\_detruire  
     liste.h, [18](#)  
 Liste\_egal  
     liste.h, [18](#)  
 Liste\_insererAIndex  
     liste.h, [19](#)  
 Liste\_insererDebut  
     liste.h, [19](#)  
 Liste\_insererFin  
     liste.h, [19](#)  
 Liste\_insererListeAIndex  
     liste.h, [19](#)  
 Liste\_insererListeDebut  
     liste.h, [20](#)  
 Liste\_insererListeFin  
     liste.h, [20](#)  
 Liste\_premierElement  
     liste.h, [20](#)  
 Liste\_sousListe  
     liste.h, [20](#)  
 Liste\_struct, [6](#)  
     a, [7](#)  
     afficher, [7](#)  
     dernierElement, [7](#)  
     egal, [7](#)  
     insererAIndex, [7](#)  
     insererDebut, [8](#)  
     insererFin, [8](#)  
     insererListeAIndex, [8](#)  
     insererListeDebut, [8](#)  
     insererListeFin, [8](#)  
     premierElement, [8](#)  
     sousListe, [8](#)  
     supprimerAIndex, [9](#)  
     supprimerDernier, [9](#)  
     supprimerPremier, [9](#)  
     trier, [9](#)  
     vider, [9](#)  
 Liste\_supprimerAIndex  
     liste.h, [20](#)  
 Liste\_supprimerDernier  
     liste.h, [21](#)  
 Liste\_supprimerPremier  
     liste.h, [21](#)  
 Liste\_trier  
     liste.h, [21](#)  
 Liste\_vider  
     liste.h, [21](#)  
 premierElement  
     Liste\_struct, [8](#)  
 sousListe  
     Liste\_struct, [8](#)  
 supprimerAIndex  
     Liste\_struct, [9](#)  
 supprimerDernier  
     Liste\_struct, [9](#)  
 supprimerPremier  
     Liste\_struct, [9](#)  
 trier  
     Liste\_struct, [9](#)  
 typeAssocie, [9](#)  
     fonctionAffichage, [10](#)  
     fonctionCompare, [10](#)  
 typeAssocie\_creer  
     liste.h, [22](#)  
 typeAssocie\_estEgal  
     liste.h, [22](#)  
 typeAssocie\_estNull  
     liste.h, [22](#)  
 vider  
     Liste\_struct, [9](#)