Liste doublement chainée

Généré par Doxygen 1.8.2

Samedi Décembre 1 2012 23 :16 :52

Table des matières

Chapitre 1

Index des structures de données

1.1 Structures de données

Liste des structures de données avec une brève description :

Chapitre 2

Index des fichiers

2.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

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 précedente.
- 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 peux
      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
  Mémorise 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 ∗lAInserer)
  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 ∗lAInserer)
  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 ∗lAInserer, 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 chainée.

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_insererDebut(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) -> l.a(l,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 ∗lAInserer, 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 ∗lAInserer)

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 ∗lAInserer)

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, extrai 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 dejà 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 personalisés 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 lAInserer)
  Insere une liste au début d'une autre liste.
- int Liste_insererListeFin (Liste l, Liste lAInserer)
  Insere une liste à la fin d'une autre liste.
- int Liste_insererListeAIndex (Liste l, Liste lAInserer, 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 contedu 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 associes.
- 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.

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

ch	la chaine à afficher.

4.1.3.2 int __CHAINE__compare (const void ∗ *ch1,* const void ∗ *ch2*)

Comparer 2 chaines de caractères.

Paramètres

ch1,ch2	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

ch	le char à afficher.

4.1.3.4 int __CHAR__compare (const void * *ch1*, const void * *ch2*)

Comparer 2 char.

Paramètres

ch1,ch2	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

dou	le double à afficher.

4.1.3.6 int __DOUBLE__compare (const void * *dou1*, const void * *dou2*)

Compare 2 double.

Paramètres

dou1,dou2	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

fl	le float à afficher.

4.1.3.8 int __FLOAT__compare (const void * *fl1*, const void * *fl2*)

Compare 2 float.

Paramètres

fl1,fl2	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

in	le int à afficher.

4.1.3.10 int __INT__compare (const void * *in1,* const void * *in2*)

Compare 2 int.

Paramètres

in1,in2	2 int à comparer.

Renvoie

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

4.1.3.11 void __LONG__affiche (void * *lo*)

Affiche le contenu d'un long.

Paramètres

lo	le long à afficher.

4.1.3.12 int __LONG__compare (const void * *lo1,* const void * *lo2*)

Comparer 2 long.

Paramètres

lo1,lo2	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

sh	le short à afficher

4.1.3.14 int __SHORT__compare (const void * *sh1,* const void * *sh2*)

Comparer 2 short.

Paramètres

sh1,sh2	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

c	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

c	la cellule à copier.

4.1.3.17 int Cellule_comparer (Cellule *c1,* Cellule *c2*)

Compare 2 cellules.

Paramètres

c1,c2	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

data	l'information a insérer dans la cellule.

Renvoie

le pointeur vers la cellule crée.

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 detruire.

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 contedu d'une liste.

Paramètres

l	la liste a afficher.

4.1.3.24 Liste Liste_cloner (Liste *l*)

Copie une liste.

Renvoie

le pointeur vers la copie de la liste.

Paramètres

l	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 typeAssocie à la liste.

Renvoie

le pointeur vers la liste crée.

4.1.3.26 Cellule Liste_dernierElement (Liste *l*)

Accès au dernier élément de la liste.

Paramètres

l	la liste.

Renvoie

Pointeur vers la dernière cellule.

4.1.3.27 int Liste_detruire (Liste *l*)

Détruit une liste.

Paramètres

l	la liste à détruire.

Renvoie

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

4.1.3.28 int Liste_egal (Liste *l1*, Liste *l2*)

Teste l'égalité deux listes.

Paramètres

l1,l2	deux listes à tester.

Renvoie

1 si deux listes sont égaux, sinon 0.

4.1.3.29 int Liste_insererAIndex (Liste *l,* Cellule *c,* int *index*)

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

Paramètres

l	la liste.
c	la cellule à insérer dans la liste.
index	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

l	la liste.
c	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

l	la liste.
c	la cellule à insérer dans la liste.

Renvoie

-1 si erreur, 0 si succès.

4.1.3.32 int Liste_insererListeAIndex (Liste *l,* Liste *lAInserer,* int *index*)

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

Paramètres

l	la liste où on insère.
lAInserer	la liste à insérer.
index	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

l	la liste où on insère.
lAInserer	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

l	la liste où on insère.
lAInserer	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

l	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

l	la liste.
index	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

l	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 *l*)

Supprime le dernier élément d'une liste.

Paramètres

l	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 *l*)

Supprime le 1er élément d'une liste.

Paramètres

l	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 *l*)

Trie une liste.

Paramètres

l	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 *l*)

Vide une liste.

Paramètres

l	la liste à vider.

Renvoie

0 si succès, -1 si la liste est null.

4.1.3.42 typeAssocie typeAssocie_creer (void *fctAffichagevoid* ∗, int *fctCompareconst void* ∗, *const void* ∗)

Crée un type associé.

Paramètres

fctAffichage	(void ∗) la fonction affichage propre du type associé.
fctCompare	(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

ta1,ta2	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

ta	type associé à tester.

Renvoie

1 si nul, sinon 0.

Index