

Ruzzle Solver

Généré par Doxygen 1.8.10

Lundi 23 Novembre 2015 21 :43 :02

Table des matières

1	Page principale	1
2	Index des structures de données	3
2.1	Structures de données	3
3	Index des fichiers	5
3.1	Liste des fichiers	5
4	Documentation des structures de données	7
4.1	Référence de la structure cell	7
4.1.1	Description détaillée	7
4.1.2	Documentation des champs	7
4.1.2.1	bonus	8
4.1.2.2	character	8
4.1.2.3	point	8
4.1.2.4	visited	8
4.2	Référence de la structure coord	8
4.2.1	Description détaillée	8
4.2.2	Documentation des champs	9
4.2.2.1	x	9
4.2.2.2	y	9
4.3	Référence de la structure element	9
4.3.1	Description détaillée	9
4.3.2	Documentation des champs	9
4.3.2.1	pred	9
4.3.2.2	score	10
4.3.2.3	succ	10
4.3.2.4	word	10
4.4	Référence de la structure t_score	10
4.4.1	Description détaillée	10
4.4.2	Documentation des champs	10
4.4.2.1	multi	11

4.4.2.2	point	11
5	Documentation des fichiers	13
5.1	Référence du fichier liste.c	13
5.1.1	Description détaillée	14
5.1.2	Documentation des définitions de type	14
5.1.2.1	t_element	14
5.1.3	Documentation des fonctions	14
5.1.3.1	ajout_droit(char *inWord, int score)	14
5.1.3.2	ajout_gauche(char *inWord, int score)	15
5.1.3.3	en_queue(void)	15
5.1.3.4	en_tete(void)	16
5.1.3.5	hors_liste(void)	16
5.1.3.6	init_liste(void)	16
5.1.3.7	liste_vide(void)	16
5.1.3.8	suivant(void)	17
5.1.3.9	valeur_elt(char mot[20], int *score)	17
5.1.4	Documentation des variables	17
5.1.4.1	drapeau	17
5.1.4.2	ec	17
5.2	Référence du fichier liste.h	18
5.2.1	Documentation des fonctions	18
5.2.1.1	ajout_droit(char *, int)	18
5.2.1.2	ajout_gauche(char *, int)	19
5.2.1.3	en_queue(void)	19
5.2.1.4	en_tete(void)	20
5.2.1.5	hors_liste(void)	20
5.2.1.6	init_liste(void)	20
5.2.1.7	liste_vide(void)	20
5.2.1.8	suivant(void)	21
5.2.1.9	valeur_elt(char[], int *)	21
5.3	Référence du fichier main.c	21
5.3.1	Description détaillée	22
5.3.2	Documentation des macros	22
5.3.2.1	N	22
5.3.3	Documentation des fonctions	22
5.3.3.1	main(int argc, char *argv[])	22
5.4	Référence du fichier README.md	22
5.5	Référence du fichier ruzzle.c	23
5.5.1	Description détaillée	23

5.5.2	Documentation des fonctions	24
5.5.2.1	add_score(cell ruzzle[N][N], coord pos, t_score *score)	24
5.5.2.2	add_word(char *mot, int score)	24
5.5.2.3	create_new_dico(cell ruzzle[N][N], FILE *dico)	25
5.5.2.4	find_letter(char c, coord *pos, cell ruzzle[N][N])	25
5.5.2.5	find_word(cell ruzzle[N][N], char mot[])	26
5.5.2.6	init(char entree[], cell sortie[N][N], FILE *dico, FILE *output)	26
5.5.2.7	init_visited(cell ruzzle[N][N])	27
5.5.2.8	print_list()	27
5.6	Référence du fichier ruzzle.h	27
5.6.1	Documentation des macros	29
5.6.1.1	N	29
5.6.2	Documentation du type de l'énumération	29
5.6.2.1	bonusEnum	29
5.6.3	Documentation des fonctions	29
5.6.3.1	ajouter_mot(char *, int)	29
5.6.3.2	ajouter_score(cell[N][N], coord, t_score *)	29
5.6.3.3	create_new_dico(cell[N][N], FILE *)	29
5.6.3.4	find_letter(char, coord *, cell[N][N])	30
5.6.3.5	find_word(cell[N][N], char[])	30
5.6.3.6	init(char[], cell[N][N], FILE *, FILE *)	31
5.6.3.7	init_visited(cell[N][N])	31
5.6.3.8	print_list()	32
Index		33

Chapitre 1

Page principale

Ce programme à été conçue dans le cadre d'un projet de seconde année de License SPI à l'Université du Maine - Le Mans

Ce programme à pour but de résoudre une grille du jeux pour smartphone Ruzzle :

Il est possible d'utilisé ce programme soit :

- En CLI en passant directement la chaine de caractère qui défini la grille

```
> ruzzle_solver "V5 R1 L2 S1 T1 N1 I1 V5 C3MTL2 A1 R1 D2 E1 S1 E1 "
```

- En entrent la chaine de caractère lorsqu'elle est demandée.

Auteurs : Ewen C. - Bastien

Chapitre 2

Index des structures de données

2.1 Structures de données

Liste des structures de données avec une brève description :

cell	Défini une case de la grille, l'élément 'visited' est utilisé lors de la recherche de mot	??
coord	Défini des coordonnées d'une case de la grille	??
element	Structure de donnée formant un élément de la liste	??
t_score	Défini le score du mot que l'on est en train de chercher	??

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

liste.c	Programme de gestion d'une liste	??
liste.h	??
main.c	Programme principal	??
ruzzle.c	Fonctions du ruzzle solver	??
ruzzle.h	??

Chapitre 4

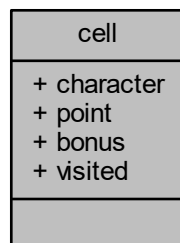
Documentation des structures de données

4.1 Référence de la structure cell

Défini une case de la grille, l'élément 'visited' est utilisé lors de la recherche de mot.

```
#include <ruzzle.h>
```

Graphe de collaboration de cell :



Champs de données

- unsigned char [character](#)
- unsigned int [point](#)
- [bonusEnum](#) bonus
- unsigned int [visited](#)

4.1.1 Description détaillée

Défini une case de la grille, l'élément 'visited' est utilisé lors de la recherche de mot.

Définition à la ligne 23 du fichier ruzzle.h.

4.1.2 Documentation des champs

4.1.2.1 `bonusEnum bonus`

Définition à la ligne 27 du fichier `ruzzle.h`.

4.1.2.2 `unsigned char character`

Définition à la ligne 25 du fichier `ruzzle.h`.

4.1.2.3 `unsigned int point`

Définition à la ligne 26 du fichier `ruzzle.h`.

4.1.2.4 `unsigned int visited`

Définition à la ligne 28 du fichier `ruzzle.h`.

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

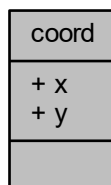
— [ruzzle.h](#)

4.2 Référence de la structure `coord`

Défini des coordonnées d'une case de la grille.

```
#include <ruzzle.h>
```

Graphe de collaboration de `coord` :



Champs de données

- `int x`
- `int y`

4.2.1 Description détaillée

Défini des coordonnées d'une case de la grille.

Définition à la ligne 36 du fichier `ruzzle.h`.

4.2.2 Documentation des champs

4.2.2.1 int x

Définition à la ligne 38 du fichier ruzzle.h.

4.2.2.2 int y

Définition à la ligne 39 du fichier ruzzle.h.

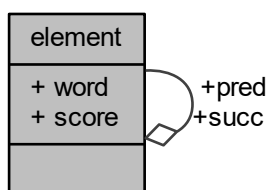
La documentation de cette structure a été générée à partir du fichier suivant :

— [ruzzle.h](#)

4.3 Référence de la structure element

Structure de donnée formant un élément de la liste.

Graphe de collaboration de element :



Champs de données

- char [word](#) [20]
- int [score](#)
- struct [element](#) * [pred](#)
- struct [element](#) * [succ](#)

4.3.1 Description détaillée

Structure de donnée formant un élément de la liste.

Définition à la ligne 24 du fichier liste.c.

4.3.2 Documentation des champs

4.3.2.1 struct element* pred

Pointeur vers l'élément précédent

Définition à la ligne 28 du fichier liste.c.

4.3.2.2 int score

Score enregistré dans l'élément de la chaîne

Définition à la ligne 27 du fichier liste.c.

4.3.2.3 struct element* succ

Pointeur vers l'élément suivant

Définition à la ligne 29 du fichier liste.c.

4.3.2.4 char word[20]

Chaîne de caractère enregistré dans l'élément de la chaîne

Définition à la ligne 26 du fichier liste.c.

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

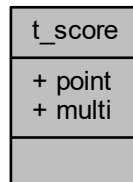
— [liste.c](#)

4.4 Référence de la structure t_score

défini le score du mot que l'on est en train de chercher.

```
#include <ruzzle.h>
```

Graphe de collaboration de t_score :



Champs de données

- int [point](#)
- int [multi](#)

4.4.1 Description détaillée

défini le score du mot que l'on est en train de chercher.

Définition à la ligne 47 du fichier ruzzle.h.

4.4.2 Documentation des champs

4.4.2.1 `int multi`

Définition à la ligne 50 du fichier `ruzzle.h`.

4.4.2.2 `int point`

Définition à la ligne 49 du fichier `ruzzle.h`.

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

— [ruzzle.h](#)

Chapitre 5

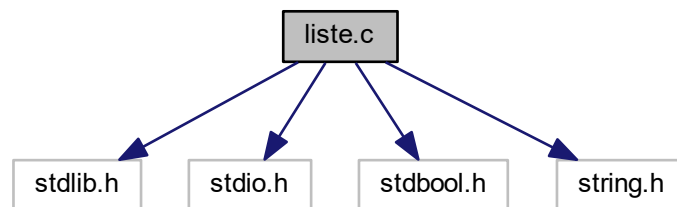
Documentation des fichiers

5.1 Référence du fichier liste.c

Programme de gestion d'une liste.

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

Graphe des dépendances par inclusion de liste.c :



Structures de données

- struct `element`
Structure de donnée formant un élément de la liste.

Définitions de type

- typedef struct `element` `t_element`
Structure de donnée formant un élément de la liste.

Fonctions

- void `init_liste` (void)
Initialise la liste à vide.
- bool `liste_vide` (void)
Fonction permettant de savoir si la liste contient au moins un element.

- bool `hors_liste` (void)
Verifie si on se trouve encore dans la liste.
- void `en_tete` (void)
Positionne en tete de la liste.
- void `en_queue` (void)
Positionne en queue de la liste.
- void `suivant` (void)
Positionne sur l'element suivant.
- void `valeur_elt` (char mot[20], int *score)
Renvoie les valeurs comprises dans l'element courant.
- void `ajout_droit` (char *inWord, int score)
Ajoute un nouvel élément a droite de l'elt courant.
- void `ajout_gauche` (char *inWord, int score)
Ajoute un nouvel élément a gauche de l'elt courant.

Variables

- `t_element` * `drapeau`
- `t_element` * `ec`

5.1.1 Description détaillée

Programme de gestion d'une liste.

Auteur

Ewen C. Bastien B.

Version

1.0

Date

22/11/2015

Gestion d'une liste doublement chaînée contenant une chaîne de caractère et un entier

5.1.2 Documentation des définitions de type

5.1.2.1 typedef struct element t_element

Structure de donnée formant un élément de la liste.

5.1.3 Documentation des fonctions

5.1.3.1 void ajout_droit (char * inWord, int score)

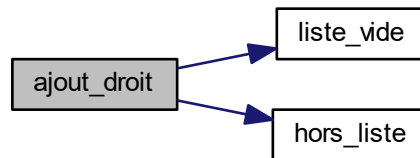
Ajoute un nouvel élément a droite de l'elt courant.

Paramètres

<i>inWord</i>	: Mot à ajouter
<i>score</i>	: Score à ajouter

Définition à la ligne 129 du fichier liste.c.

Voici le graphe d'appel pour cette fonction :



5.1.3.2 void ajout_gauche (char * *inWord*, int *score*)

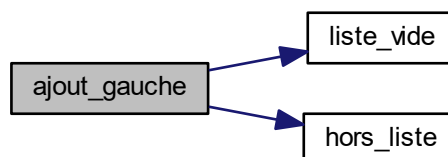
Ajoute un nouvel élément a gauche de l'elt courant.

Paramètres

<i>inWord</i>	: Mot à ajouter
<i>score</i>	: Score à ajouter

Définition à la ligne 153 du fichier liste.c.

Voici le graphe d'appel pour cette fonction :



5.1.3.3 void en_queue (void)

Positionne en queue de la liste.

Définition à la ligne 89 du fichier liste.c.

Voici le graphe d'appel pour cette fonction :



5.1.3.4 void en_tete (void)

Positionne en tete de la liste.

Définition à la ligne 78 du fichier liste.c.

Voici le graphe d'appel pour cette fonction :



5.1.3.5 bool hors_liste (void)

Verifie si on se trouve encore dans la liste.

Renvoie

Renvoie un booléen à vrai si l'element courant est hors de la liste, faux sinon.

Définition à la ligne 68 du fichier liste.c.

5.1.3.6 void init_liste (void)

Initialise la liste à vide.

Définition à la ligne 41 du fichier liste.c.

5.1.3.7 bool liste_vide (void)

Fonction permettant de savoir si la liste contient au moins un element.

Renvoie

Renvoie un booléen à vrai si la liste est vide, faux sinon.

Définition à la ligne 56 du fichier liste.c.

5.1.3.8 void suivant (void)

Positionne sur l'element suivant.

Définition à la ligne 100 du fichier liste.c.

Voici le graphe d'appel pour cette fonction :



5.1.3.9 void valeur_elt (char *mot*[20], int * *score*)

Renvoie les valeurs comprises dans l'element courant.

Paramètres

<i>mot</i>	: Le mot enregistré dans la liste
<i>score</i>	: Le score enregistré dans la liste

Définition à la ligne 114 du fichier liste.c.

Voici le graphe d'appel pour cette fonction :



5.1.4 Documentation des variables

5.1.4.1 t_element* drapeau

Pointeur sur l'élément initial de la liste.

Définition à la ligne 33 du fichier liste.c.

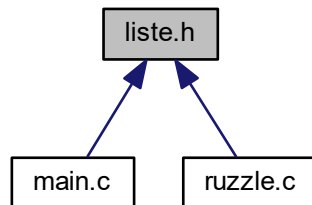
5.1.4.2 t_element* ec

Pointeur sur l'élément courant de la liste.

Définition à la ligne 34 du fichier liste.c.

5.2 Référence du fichier liste.h

Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Fonctions

- void `init_liste` (void)
Initialise la liste à vide.
- bool `liste_vide` (void)
Fonction permettant de savoir si la liste contient au moins un element.
- bool `hors_liste` (void)
Verifie si on se trouve encore dans la liste.
- void `en_tete` (void)
Positionne en tete de la liste.
- void `en_queue` (void)
Positionne en queue de la liste.
- void `suivant` (void)
Positionne sur l'element suivant.
- void `valeur_elt` (char[], int *)
- void `ajout_droit` (char *, int)
Ajoute un nouvel élément a droite de l'elt courant.
- void `ajout_gauche` (char *, int)
Ajoute un nouvel élément a gauche de l'elt courant.

5.2.1 Documentation des fonctions

5.2.1.1 void ajout_droit (char * *inWord*, int *score*)

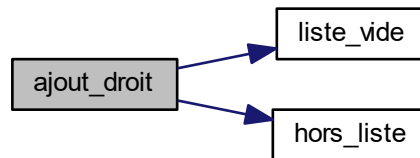
Ajoute un nouvel élément a droite de l'elt courant.

Paramètres

<i>inWord</i>	: Mot à ajouter
<i>score</i>	: Score à ajouter

Définition à la ligne 129 du fichier liste.c.

Voici le graphe d'appel pour cette fonction :



5.2.1.2 void ajout_gauche (char * *inWord*, int *score*)

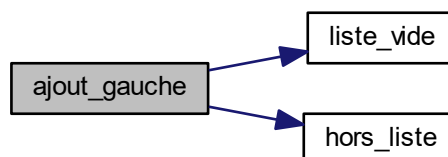
Ajoute un nouvel élément a gauche de l'elt courant.

Paramètres

<i>inWord</i>	: Mot à ajouter
<i>score</i>	: Score à ajouter

Définition à la ligne 153 du fichier liste.c.

Voici le graphe d'appel pour cette fonction :



5.2.1.3 void en_queue (void)

Positionne en queue de la liste.

Définition à la ligne 89 du fichier liste.c.

Voici le graphe d'appel pour cette fonction :



5.2.1.4 void en_tete (void)

Positionne en tete de la liste.

Définition à la ligne 78 du fichier liste.c.

Voici le graphe d'appel pour cette fonction :



5.2.1.5 bool hors_liste (void)

Verifie si on se trouve encore dans la liste.

Renvoie

Renvoi un booléen à vrai si l'element courant est hors de la liste, faux sinon.

Définition à la ligne 68 du fichier liste.c.

5.2.1.6 void init_liste (void)

Initialise la liste à vide.

Définition à la ligne 41 du fichier liste.c.

5.2.1.7 bool liste_vide (void)

Fonction permettant de savoir si la liste contient au moins un element.

Renvoie

Renvoi un booléen à vrai si la liste est vide, faux sinon.

Définition à la ligne 56 du fichier liste.c.

5.2.1.8 void suivant (void)

Positionne sur l'element suivant.

Définition à la ligne 100 du fichier liste.c.

Voici le graphe d'appel pour cette fonction :



5.2.1.9 void valeur_elt (char [], int *)

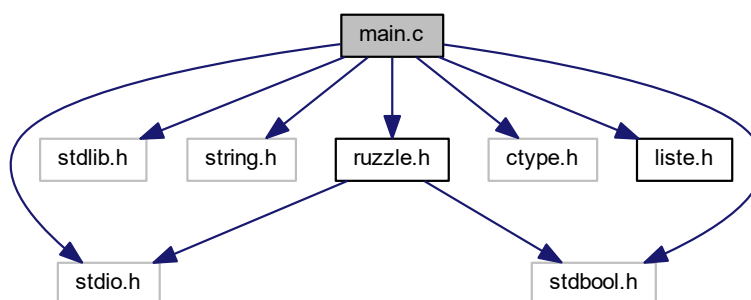
5.3 Référence du fichier main.c

Programme principal.

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <ctype.h>
#include "liste.h"
#include "ruzzle.h"
  
```

Graphe des dépendances par inclusion de main.c :



Macros

— #define N 4

Fonctions

— int main (int argc, char *argv[])

5.3.1 Description détaillée

Programme principal.

Auteur

Ewen C. Bastien B.

Version

1.0

Date

22/11/2015

Programme initialisant les structure et fichiers et appelant les fonctions filles.

5.3.2 Documentation des macros

5.3.2.1 #define N 4

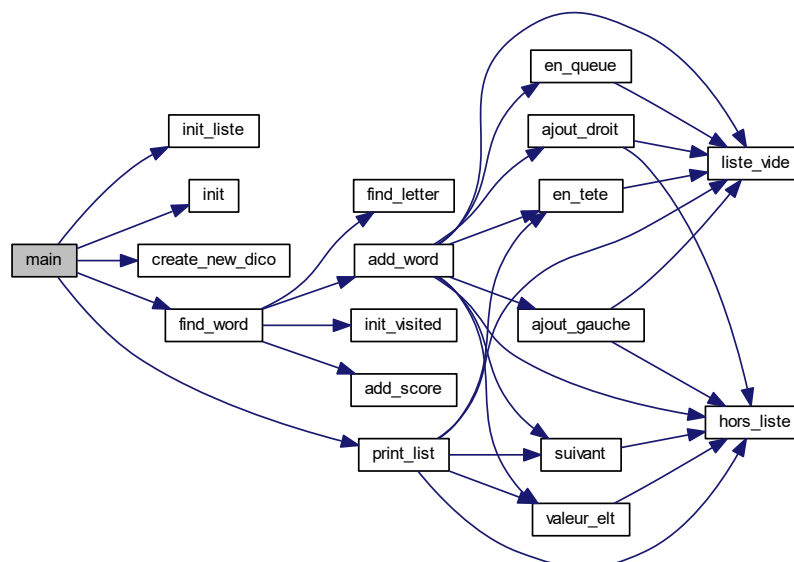
Définition à la ligne 21 du fichier main.c.

5.3.3 Documentation des fonctions

5.3.3.1 int main (int argc, char * argv[])

Définition à la ligne 23 du fichier main.c.

Voici le graphe d'appel pour cette fonction :



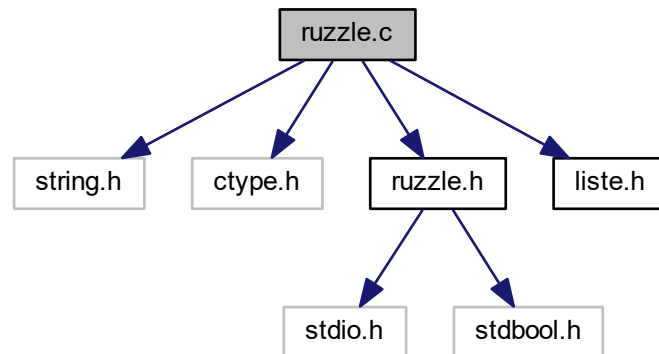
5.4 Référence du fichier README.md

5.5 Référence du fichier ruzzle.c

Fonctions du ruzzle solver.

```
#include <string.h>
#include <ctype.h>
#include "ruzzle.h"
#include "liste.h"
```

Graphe des dépendances par inclusion de ruzzle.c :



Fonctions

- int `init` (char entree[], cell sortie[N][N], FILE *dico, FILE *output)
Fonction d'initialisation globale.
- int `create_new_dico` (cell ruzzle[N][N], FILE *dico)
Crée un dictionnaire élagué
- bool `find_letter` (char c, coord *pos, cell ruzzle[N][N])
Trouve la lettre autour de la position passée en paramètre.
- void `add_word` (char *mot, int score)
Ajoute un mot et son score dans une liste triée.
- void `add_score` (cell ruzzle[N][N], coord pos, t_score *score)
Ajoute le score de la lettre courante.
- void `init_visited` (cell ruzzle[N][N])
Réinitialise le statut visité de chaque case.
- void `find_word` (cell ruzzle[N][N], char mot[])
Cherche un mot dans la grille.
- void `print_list` ()
Enregistre la liste des mots possible dans un fichier externe.

5.5.1 Description détaillée

Fonctions du ruzzle solver.

Auteur

Ewen C. Bastien B.

Version

1.0

Date

22/11/2015

Les différentes fonction nécessaire à la résolution d'une grille de ruzzle

5.5.2 Documentation des fonctions

5.5.2.1 void add_score (cell ruzzle[N][N], coord pos, t_score * score)

Ajoute le score de la lettre courante.

Paramètres

<i>cell</i>	ruzzle[N][N] grille du ruzzle
<i>coord</i>	pos position actuel
<i>t_score</i>	* score pointeur sur le score du mot que l'on cherche

ajouter_score ajoute le score de la lettre qui se trouve dans la grille a la position de type coord en parametre et l'ajoute au score (comptabilise également les multiplicateurs de mots).

Définition à la ligne 238 du fichier ruzzle.c.

5.5.2.2 void add_word (char * mot, int score)

Ajoute un mot et son score dans une liste triée.

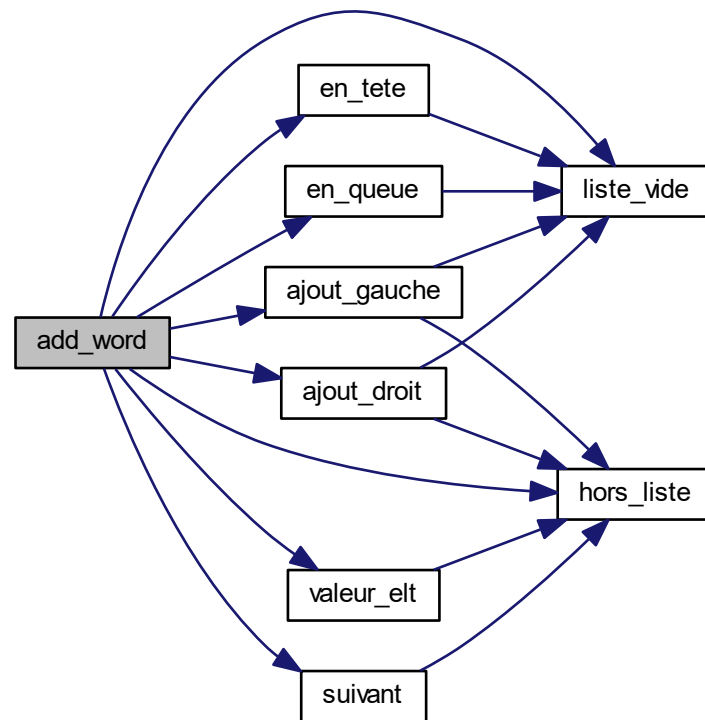
Paramètres

<i>char</i>	* mot : mot à enregistré
<i>int</i>	score : score du mot

Ajoute le mot et son score dans une liste doublement chaînée. l'insertion est effectué de sorte à garder un ordre décroissant.

Définition à la ligne 195 du fichier ruzzle.c.

Voici le graphe d'appel pour cette fonction :



5.5.2.3 int create_new_dico (cell ruzzle[N][N], FILE * dico)

Crée un dictionnaire élagué

Paramètres

<i>cell</i>	ruzzle[N][N] : matrice du ruzzle
<i>FILE</i>	* dico : dictionnaire complet

Renvoie

renvoie toujours 0 (pourrait être amélioré en renvoyant un code d'erreur)

Crée un nouveau dictionnaire appelé nexdico.txt qui ne contient que les mots dont les lettres sont présentes dans la grille. Cette fonction permet de réduire la taille du fichier parcouru de 3.6Mo à environ 800Ko.

Définition à la ligne 101 du fichier ruzzle.c.

5.5.2.4 bool find_letter (char c, coord * pos, cell ruzzle[N][N])

Trouve la lettre autour de la position passée en paramètre.

Paramètres

<i>char</i>	c caractère recherché
<i>coord</i>	* pos position autour de laquel chercher
<i>cell</i>	ruzzle[N][N] : grille du ruzzle

Renvoie

retourne true si la lettre à été trouvé, false sinon.

Définition à la ligne 159 du fichier ruzzle.c.

5.5.2.5 void find_word (cell ruzzle[N][N], char mot[])

Cherche un mot dans la grille.

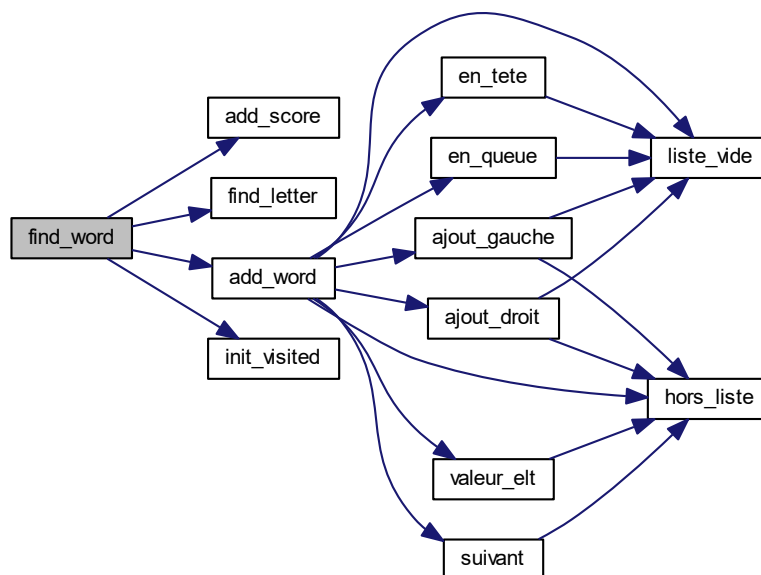
Paramètres

<i>cell</i>	ruzzle[N][N] grille du ruzzle
<i>char</i>	mot[] un mot du dictionnaire à chercher dans la grille

find_word prend en paramètre la grille du ruzzle et un mot, cherche le mot a l'aide d'autre fonction et l'inscrit dans la liste des solution avec son score.

Définition à la ligne 285 du fichier ruzzle.c.

Voici le graphe d'appel pour cette fonction :



5.5.2.6 int init (char entree[], cell sortie[N][N], FILE * dico, FILE * output)

Fonction d'initialisation globale.

Paramètres

<i>char</i>	entree[] les lettres de la grille du ruzzle.
<i>cell</i>	sortie[N][N] la transformé en tableau de entree[].
<i>FILE</i>	* dico le dictionnaire complet
<i>FILE</i>	* output le fichier où seront placé les mot possible.

Renvoie

Renvoie un code d'erreur si besoin, 0 sinon.

Initialise la grille avec les lettre passées en paramètre et verifie que les fichiers soient accessibles.

Définition à la ligne 29 du fichier ruzzle.c.

5.5.2.7 void init_visited (cell ruzzle[N][N])

Réinitialise le statut visité de chaque case.

Paramètres

<i>cell</i>	ruzzle[N][N] grille du ruzzle
-------------	-------------------------------

init_visit prend la grille du ruzzle en paramètre et réinitialise le booléen qui indique le passage ou non sur une case en le passant à false.

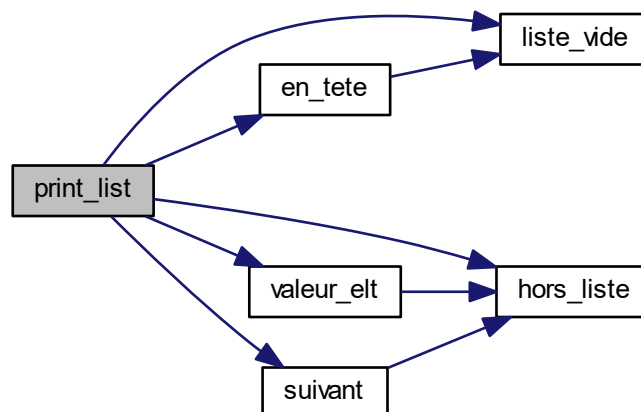
Définition à la ligne 261 du fichier ruzzle.c.

5.5.2.8 void print_list ()

Enregistre la liste des mots possible dans un fichier externe.

Définition à la ligne 337 du fichier ruzzle.c.

Voici le graphe d'appel pour cette fonction :

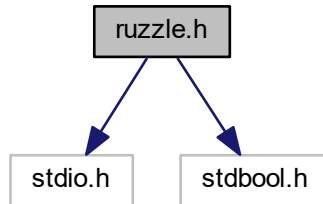


5.6 Référence du fichier ruzzle.h

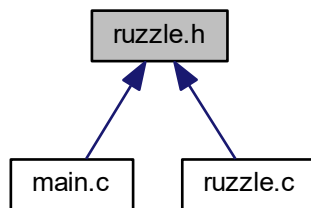
```
#include <stdio.h>
```

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

Graphe des dépendances par inclusion de ruzzle.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Structures de données

- struct `cell`
Définit une case de la grille, l'élément 'visited' est utilisé lors de la recherche de mot.
- struct `coord`
Définit des coordonnées d'une case de la grille.
- struct `t_score`
Définit le score du mot que l'on est en train de chercher.

Macros

- `#define N 4`

Énumérations

- enum `bonusEnum` {
`none = 1, doubleL = 2, tripleL = 3, doubleM = 4,`
`tripleM = 6` }
Énumération des différents bonus possibles.

Fonctions

- int `init` (char[], `cell[N][N]`, FILE *, FILE *)
Fonction d'initialisation globale.
- int `create_new_dico` (`cell[N][N]`, FILE *)
Crée un dictionnaire élagué
- bool `find_letter` (char, `coord` *, `cell[N][N]`)
Trouve la lettre autour de la position passée en paramètre.
- void `ajouter_mot` (char *, int)
- void `ajouter_score` (`cell[N][N]`, `coord`, `t_score` *)
- void `init_visited` (`cell[N][N]`)
Réinitialise le statut visité de chaque case.
- void `find_word` (`cell[N][N]`, char[])
Cherche un mot dans la grille.
- void `print_list` ()
Enregistre la liste des mots possible dans un fichier externe.

5.6.1 Documentation des macros

5.6.1.1 #define N 4

Définition à la ligne 4 du fichier ruzzle.h.

5.6.2 Documentation du type de l'énumération

5.6.2.1 enum bonusEnum

Énumération des différents bonus possibles.

Valeurs énumérées

none
doubleL
tripleL
doubleM
tripleM

Définition à la ligne 9 du fichier ruzzle.h.

5.6.3 Documentation des fonctions

5.6.3.1 void ajouter_mot (char *, int)

5.6.3.2 void ajouter_score (cell [N][N], coord , t_score *)

5.6.3.3 int create_new_dico (cell ruzzle[N][N], FILE * dico)

Crée un dictionnaire élagué

Paramètres

<i>cell</i>	<code>ruzzle[N][N]</code> : matrice du ruzzle
<i>FILE</i>	* <code>dico</code> : dictionnaire complet

Renvoie

renvoie toujours 0 (pourrait être amélioré en renvoyant un code d'erreur)

Crée un nouveau dictionnaire appelé nexdico.txt qui ne contient que les mots dont les lettres sont présentes dans la grille. Cette fonction permet de réduire la taille du fichier parcouru de 3.6Mo à environ 800Ko.

Définition à la ligne 101 du fichier ruzzle.c.

5.6.3.4 bool find_letter (char *c*, coord * *pos*, cell *ruzzle*[N][N])

Trouve la lettre autour de la position passée en paramètre.

Paramètres

<i>char</i>	<i>c</i> caractère recherché
<i>coord</i>	* <i>pos</i> position autour de laquelle chercher
<i>cell</i>	<i>ruzzle</i> [N][N] : grille du ruzzle

Renvoie

retourne true si la lettre a été trouvée, false sinon.

Définition à la ligne 159 du fichier ruzzle.c.

5.6.3.5 void find_word (cell *ruzzle*[N][N], char *mot*[])

Cherche un mot dans la grille.

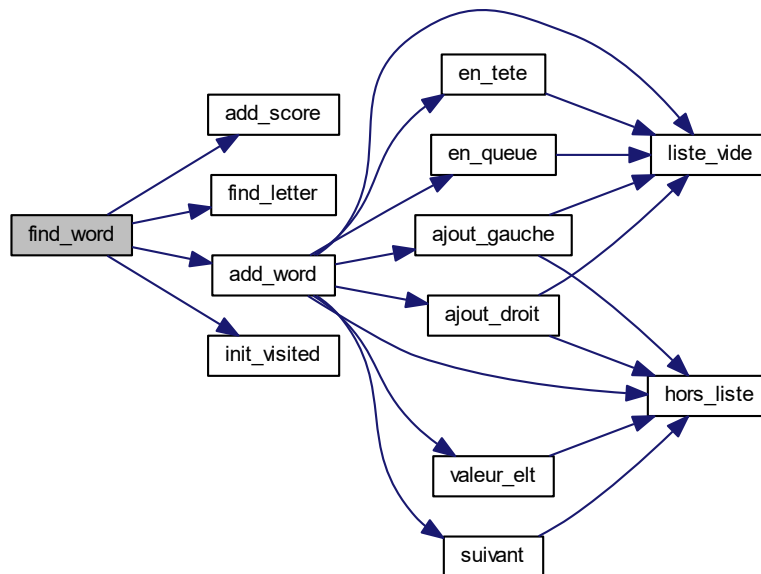
Paramètres

<i>cell</i>	<i>ruzzle</i> [N][N] grille du ruzzle
<i>char</i>	<i>mot</i> [] un mot du dictionnaire à chercher dans la grille

find_word prend en paramètre la grille du ruzzle et un mot, cherche le mot à l'aide d'autres fonctions et l'inscrit dans la liste des solutions avec son score.

Définition à la ligne 285 du fichier ruzzle.c.

Voici le graphe d'appel pour cette fonction :



5.6.3.6 int init (char entree[], cell sortie[N][N], FILE * dico, FILE * output)

Fonction d'initialisation globale.

Paramètres

<i>char</i>	entree[] les lettres de la grille du ruzzle.
<i>cell</i>	sortie[N][N] la transformé en tableau de entree[].
<i>FILE</i>	* dico le dictionnaire complet
<i>FILE</i>	* output le fichier où seront placé les mot possible.

Renvoie

Renvoie un code d'erreur si besoin, 0 sinon.

Initialise la grille avec les lettre passées en paramètre et verifie que les fichiers soient accessibles.

Définition à la ligne 29 du fichier ruzzle.c.

5.6.3.7 void init_visited (cell ruzzle[N][N])

Réinitialise le statut visité de chaque case.

Paramètres

<i>cell</i>	ruzzle[N][N] grille du ruzzle
-------------	-------------------------------

init_visit prend la grille du ruzzle en paramètre et réinitialise le booléen qui indique le passage ou non sur une case en le passant à false.

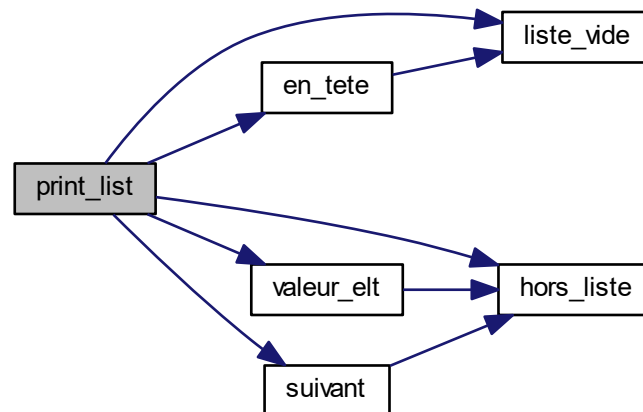
Définition à la ligne 261 du fichier ruzzle.c.

5.6.3.8 void print_list ()

Enregistre la liste des mots possible dans un fichier externe.

Définition à la ligne 337 du fichier ruzzle.c.

Voici le graphe d'appel pour cette fonction :



Index

- add_score
 - ruzzle.c, [24](#)
- add_word
 - ruzzle.c, [24](#)
- ajout_droit
 - liste.c, [14](#)
 - liste.h, [18](#)
- ajout_gauche
 - liste.c, [15](#)
 - liste.h, [19](#)
- ajouter_mot
 - ruzzle.h, [29](#)
- ajouter_score
 - ruzzle.h, [29](#)
- bonus
 - cell, [7](#)
- bonusEnum
 - ruzzle.h, [29](#)
- cell, [7](#)
 - bonus, [7](#)
 - character, [8](#)
 - point, [8](#)
 - visited, [8](#)
- character
 - cell, [8](#)
- coord, [8](#)
 - x, [9](#)
 - y, [9](#)
- create_new_dico
 - ruzzle.c, [25](#)
 - ruzzle.h, [29](#)
- doubleL
 - ruzzle.h, [29](#)
- doubleM
 - ruzzle.h, [29](#)
- drapeau
 - liste.c, [17](#)
- ec
 - liste.c, [17](#)
- element, [9](#)
 - pred, [9](#)
 - score, [9](#)
 - succ, [10](#)
 - word, [10](#)
- en_queue
 - liste.c, [15](#)
- liste.h, [19](#)
- en_tete
 - liste.c, [16](#)
 - liste.h, [20](#)
- find_letter
 - ruzzle.c, [25](#)
 - ruzzle.h, [30](#)
- find_word
 - ruzzle.c, [26](#)
 - ruzzle.h, [30](#)
- hors_liste
 - liste.c, [16](#)
 - liste.h, [20](#)
- init
 - ruzzle.c, [26](#)
 - ruzzle.h, [31](#)
- init_liste
 - liste.c, [16](#)
 - liste.h, [20](#)
- init_visited
 - ruzzle.c, [27](#)
 - ruzzle.h, [31](#)
- liste.c, [13](#)
 - ajout_droit, [14](#)
 - ajout_gauche, [15](#)
 - drapeau, [17](#)
 - ec, [17](#)
 - en_queue, [15](#)
 - en_tete, [16](#)
 - hors_liste, [16](#)
 - init_liste, [16](#)
 - liste_vide, [16](#)
 - suiwant, [16](#)
 - t_element, [14](#)
 - valeur_elt, [17](#)
- liste.h, [18](#)
 - ajout_droit, [18](#)
 - ajout_gauche, [19](#)
 - en_queue, [19](#)
 - en_tete, [20](#)
 - hors_liste, [20](#)
 - init_liste, [20](#)
 - liste_vide, [20](#)
 - suiwant, [20](#)
 - valeur_elt, [21](#)
- liste_vide

- liste.c, 16
- liste.h, 20
- main
 - main.c, 22
- main.c, 21
 - main, 22
 - N, 22
- multi
 - t_score, 10
- N
 - main.c, 22
 - ruzzle.h, 29
- none
 - ruzzle.h, 29
- point
 - cell, 8
 - t_score, 11
- pred
 - element, 9
- print_list
 - ruzzle.c, 27
 - ruzzle.h, 31
- README.md, 22
- ruzzle.c, 23
 - add_score, 24
 - add_word, 24
 - create_new_dico, 25
 - find_letter, 25
 - find_word, 26
 - init, 26
 - init_visited, 27
 - print_list, 27
- ruzzle.h, 27
 - ajouter_mot, 29
 - ajouter_score, 29
 - bonusEnum, 29
 - create_new_dico, 29
 - doubleL, 29
 - doubleM, 29
 - find_letter, 30
 - find_word, 30
 - init, 31
 - init_visited, 31
 - N, 29
 - none, 29
 - print_list, 31
 - tripleL, 29
 - tripleM, 29
- score
 - element, 9
- succ
 - element, 10
- suivant
 - liste.c, 16
 - liste.h, 20
- t_element
 - liste.c, 14
- t_score, 10
 - multi, 10
 - point, 11
- tripleL
 - ruzzle.h, 29
- tripleM
 - ruzzle.h, 29
- valeur_elt
 - liste.c, 17
 - liste.h, 21
- visited
 - cell, 8
- word
 - element, 10
- x
 - coord, 9
- y
 - coord, 9