Ruzzle Solver

Généré par Doxygen 1.8.10

Lundi 23 Novembre 2015 21 :43 :02

Table des matières

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 dirrectement 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 :

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

Chapitre 4

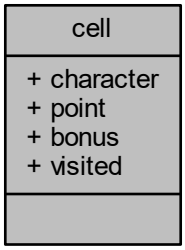
Documentation des structures de données

4.1 Référence de la structure 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.

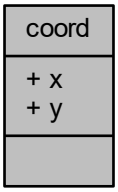
La documentation de cette structure a été générée à partir du fichier suivant :
— ruzzle.h

4.2 Référence de la structure coord

Défini des coordonnée 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 coordonnée 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

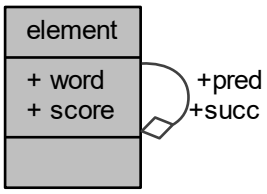
Définition à la ligne 39 du fichier ruzzle.h.

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 précedent

Définition à la ligne 28 du fichier liste.c.

4.3.2.2 int score

Score enregistré dans l'élément de la chaine

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]

Chaine de caractère enregistré dans l'élément de la chaine

Définition à la ligne 26 du fichier liste.c.

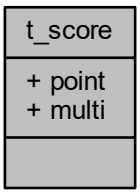
La documentation de cette structure a été générée à partir du fichier suivant :
- liste.c

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

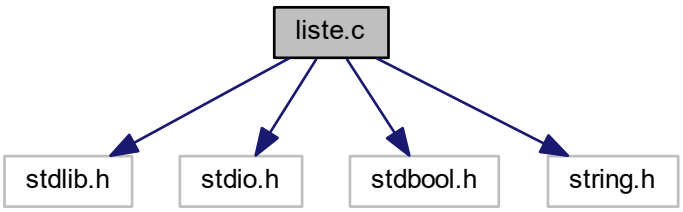
Documentation des fichiers

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)
  Renvoi 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 chainée contenant une chaine 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*)

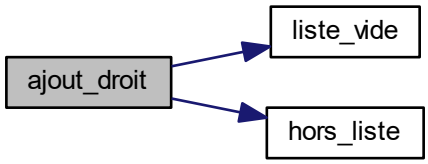
Ajoute un nouvel élément a droite de l'elt courant.

Paramètres

inWord	: Mot à ajouter
score	: 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*)

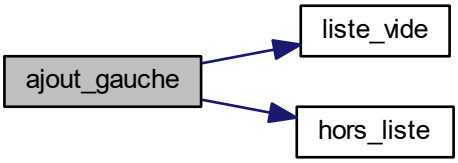
Ajoute un nouvel élément a gauche de l'elt courant.

Paramètres

inWord	: Mot à ajouter
score	: 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

Renvoi un booleen 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

Renvoi 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*)

Renvoi les valeurs comprises dans l'element courant.

Paramètres

mot	: Le mot enregistré dans la liste
score	: 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.

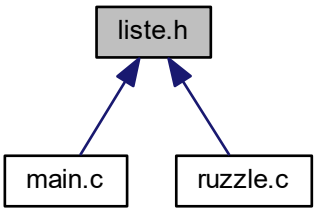
5.1.4.2 t_element* ec

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*)

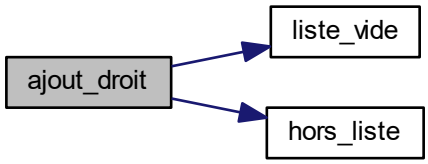
Ajoute un nouvel élément a droite de l'elt courant.

Paramètres

inWord	: Mot à ajouter
score	: 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*)

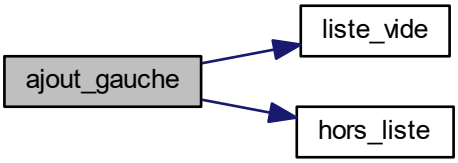
Ajoute un nouvel élément a gauche de l'elt courant.

Paramètres

inWord	: Mot à ajouter
score	: 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 booleen 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 :

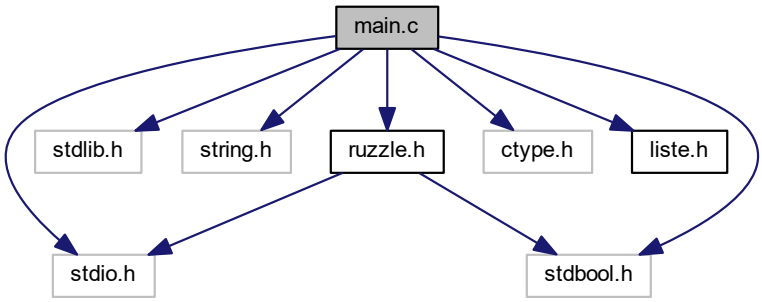
5.2.1.9 void valeur_elt (char [ ], int *)

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

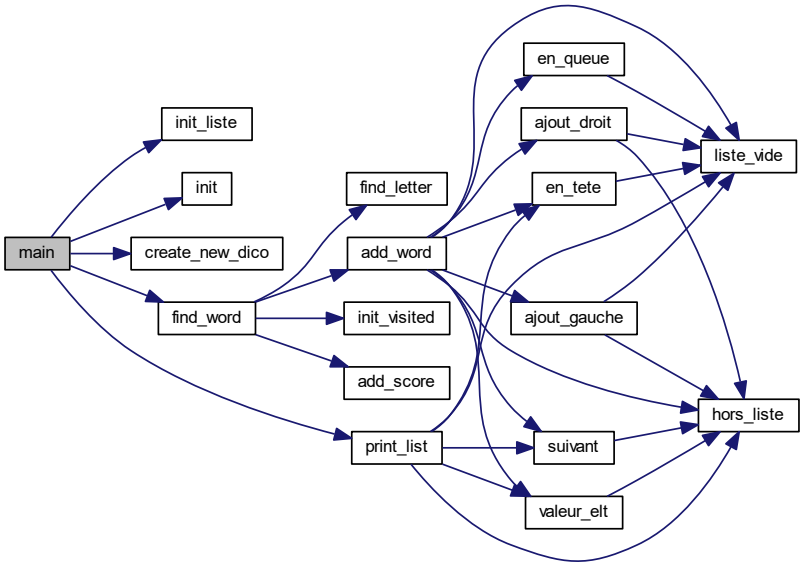
Définition à la ligne 21 du fichier main.c.

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 :



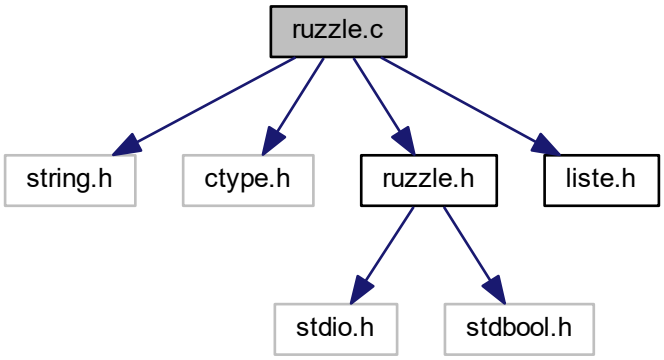
5.4 Référence du fichier README.md

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 dictionaire 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 ()
  Enregiste 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

cell	ruzzle[N][N] grille du ruzzle
coord	pos position actuel
t_score	* 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.

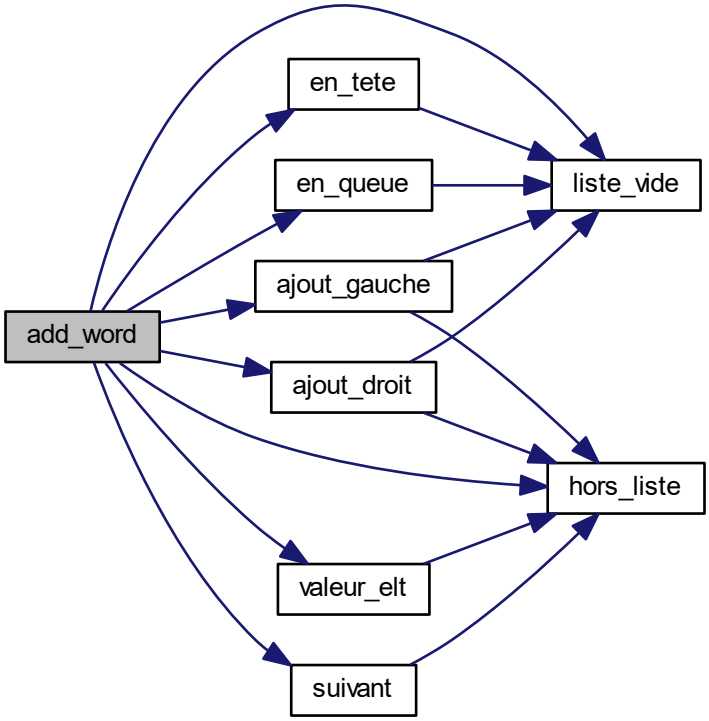
Paramètres

char	* mot : mot à enregistré
int	score : score du mot

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

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 dictionaire élagué

Paramètres

cell	ruzzle[N][N] : matrice du ruzzle
FILE	* dico : dictionnaire complet

Renvoie

renvoie toujours 0 (pourait être amélioré en renvoiyant un code d'erreur)

Crée un nouveau dictionnaire appelé nexdico.txt qui ne contiend que les mot 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

char	c charactère recherché
coord	∗ pos position autour de laquel chercher
cell	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.

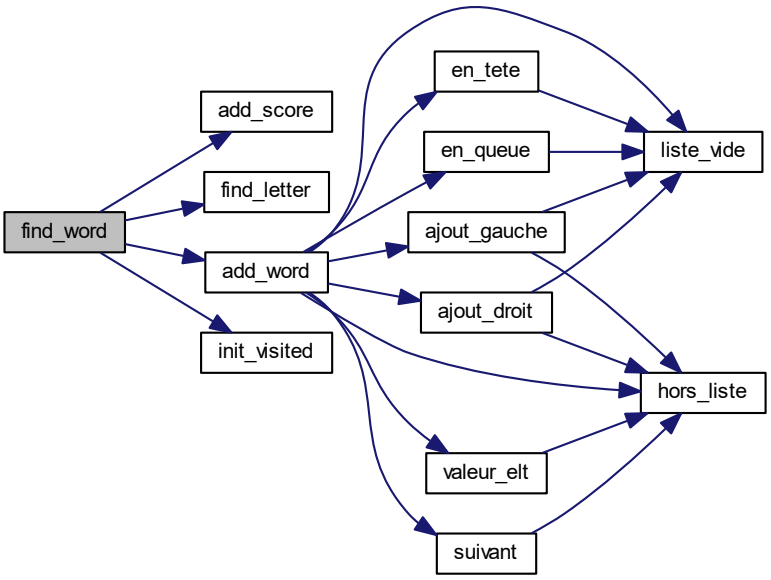
Paramètres

cell	ruzzle[N][N] grille du ruzzle
char	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

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

Renvoie

Renvoit 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

cell	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.

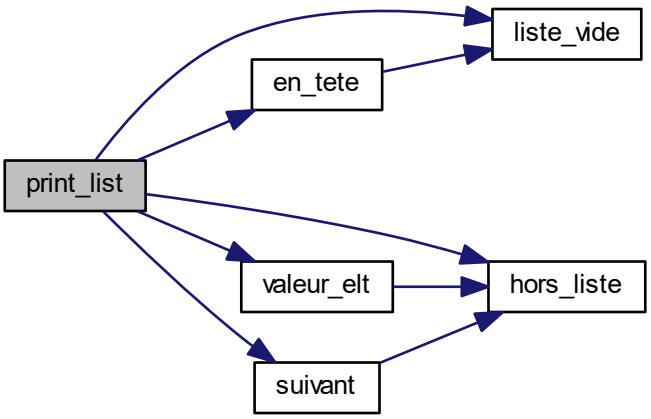
Définition à la ligne 261 du fichier ruzzle.c.

5.5.2.8 void print_list ( )

Enregiste 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 :

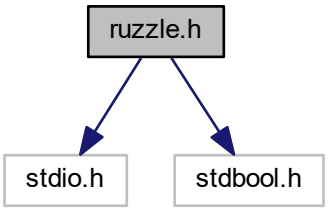


5.6 Référence du fichier ruzzle.h

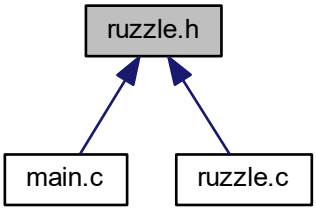
```
#include <stdio.h>
```

```
#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
  Défini une case de la grille, l'élément 'visited' est utilisé lors de la recherche de mot.
- struct coord
  Défini des coordonnée d'une case de la grille.
- struct t_score
  défini 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 dictionaire 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 ()
  Enregiste 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 dictionaire élagué

Paramètres

cell	ruzzle[N][N] : matrice du ruzzle
FILE	∗ dico : dictionnaire complet

Renvoie

renvoie toujours 0 (pourait être amélioré en renvoiyant un code d'erreur)

Crée un nouveau dictionnaire appelé nexdico.txt qui ne contiend que les mot 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

char	c charactère recherché
coord	* pos position autour de laquel chercher
cell	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.

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

Cherche un mot dans la grille.

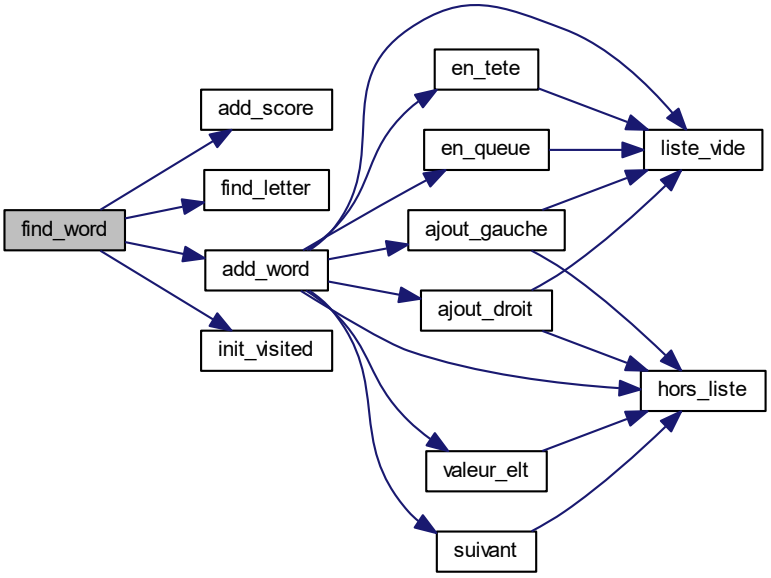
Paramètres

cell	ruzzle[N][N] grille du ruzzle
char	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 :



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

Fonction d'initialisation globale.

Paramètres

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

Renvoie

Renvoit 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

cell	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.

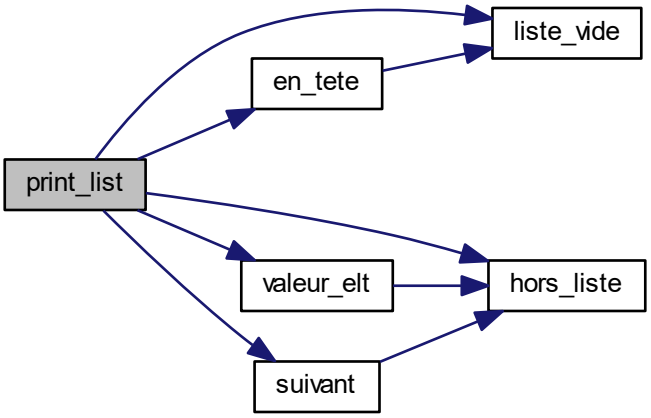
Définition à la ligne 261 du fichier ruzzle.c.

5.6.3.8 void print_list ( )

Enregiste 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