Projet_LO21

1.0

Généré par Doxygen 1.8.18

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 avec une brève description :

Chapitre 3

Documentation des structures de données

3.1 Référence de la structure BDConnaissancesElem

`#include <BDConnaissances.h>`

Champs de données

- **Regle** * **valeur**
- struct **BDConnaissancesElem** * **suivant**

3.1.1 Description détaillée

Définition à la ligne 23 du fichier BDConnaissances.h.

3.1.2 Documentation des champs

3.1.2.1 suivant

`struct` **`BDConnaissancesElem`**`* suivant`

valeur de l'élément de la liste chainée ici un pointeur sur **Regle** (p. 8)

Définition à la ligne 26 du fichier BDConnaissances.h.

3.1.2.2 valeur

`Regle``* valeur`

Définition à la ligne 25 du fichier BDConnaissances.h.

La documentation de cette structure a été générée à partir du fichier suivant :
- src/ **BDConnaissances.h**

3.2 Référence de la structure BDConnaissanceselem

un élément de la liste chainé de Règles

`#include <BDConnaissances.h>`

3.2.1 Description détaillée

un élément de la liste chainé de Règles

Auteur

Florian CLOAREC

La documentation de cette structure a été générée à partir du fichier suivant :
— src/ **BDConnaissances.h**

3.3 Référence de la structure Premisse

strucutre de donné permetant de représenter une prémisse par une liste chainé de propositon

`#include <premisse.h>`

3.3.1 Description détaillée

strucutre de donné permetant de représenter une prémisse par une liste chainé de propositon

Auteur

Florian CLOAREC

La documentation de cette structure a été générée à partir du fichier suivant :
— src/ **premisse.h**

3.4 Référence de la structure PremisseElem

structure de donnée permetant de stocker un élément de la liste chainé de Proposion qui représente la Prémisse

`#include <premisse.h>`

Champs de données

— **Proposition** ∗ **valeur**
— struct **PremisseElem** ∗ **elemSuivant**

3.4.1 Description détaillée

structure de donnée permetant de stocker un élément de la liste chainé de Proposion qui représente la Prémisse

Auteur

Florian CLOAREC

Définition à la ligne 23 du fichier premisse.h.

3.4.2 Documentation des champs

3.4.2.1 elemSuivant

```
struct PremisseElem* elemSuivant
```

valeur de l'élément de la liste chainé

Définition à la ligne 26 du fichier premisse.h.

3.4.2.2 valeur

```
Proposition* valeur
```

Définition à la ligne 25 du fichier premisse.h.

La documentation de cette structure a été générée à partir du fichier suivant :
- src/ **premisse.h**

3.5 Référence de la structure Proposition

structure de donnée permetant de stocker ou une Propositon

```
#include <proposition.h>
```

Champs de données

- char * **description**
- bool **validite**

3.5.1 Description détaillée

structure de donnée permetant de stocker ou une Propositon

Définition à la ligne 23 du fichier proposition.h.

3.5.2 Documentation des champs

3.5.2.1 description

```
char* description
```

Définition à la ligne 25 du fichier proposition.h.

3.5.2.2 validite

```
bool validite
```

chaine de caracère qui contient la description en laguage naturel de la **Proposition** (p. 7)

Définition à la ligne 26 du fichier proposition.h.

La documentation de cette structure a été générée à partir du fichier suivant :
- src/ **proposition.h**

3.6 Référence de la structure Regle

structure de donnée qui représente une règle

```
#include <regle.h>
```

Champs de données

- **Premisse premisse**
- **Proposition * conclusion**

3.6.1 Description détaillée

structure de donnée qui représente une règle

Définition des structure liée à une Règle

Définition à la ligne 25 du fichier regle.h.

3.6.2 Documentation des champs

3.6.2.1 conclusion

```
Proposition* conclusion
```

la liste chainé contenant les **Proposition** (p. 7) qui forment la prémisse de la règle

Définition à la ligne 28 du fichier regle.h.

3.6.2.2 premisse

```
Premisse premisse
```

Définition à la ligne 27 du fichier regle.h.

La documentation de cette structure a été générée à partir du fichier suivant :
- src/ **regle.h**

Chapitre 4

Documentation des fichiers

4.1 Référence du fichier src/BDConnaissances.c

fichier contenant les implémentations des fonctions lié à la structure de donnée Base de Connaissances

```
#include "main.h"
```

Fonctions

- bool **isEmptyBDC** (**BDConnaissances** bdc)
  teste si une BDConnaissances est vide
- **BDConnaissances** **addHeadBDC** (**BDConnaissances** bdc, **Regle** ∗aAjouter)
  ajoute une règle à une base de connaissances en tête
- **BDConnaissances** **deleteHeadBDC** (**BDConnaissances** bdc)
  supprime l'élément en tête de la bdc
- void **deleteAllBDC** (**BDConnaissances** bdc)
  supprime récursivement la totalitée d'une base de connaissance
- void **afficheBDC** (**BDConnaissances** bdc)
  affiche la totalitée d'une base de connaissances
- **BDConnaissances** **addRegleBDC** (**BDConnaissances** bdc, **Premisse** ∗pListProp, char ∗description↩
  Premisse[ ], long nbElemPremisse, char ∗descriptionConclusion)
  ajoute une règle à une BDC à partir de chaine de caractère
- **Premisse** **moteurDInference** (**Premisse** baseVerite, **BDConnaissances** bdc)
  recherche à partir de la base de véritée et de la base de connaissances les propositions qui sont vraie
- **Premisse** **createBDVerite** (**Premisse** listProp, **Premisse** BDVerite)
  crée la base de connaissance en y ajoutant toutee les proposition qui sont vraie
- bool **isPropositionInConclusion** (**BDConnaissances** bdc, **Proposition** ∗prop)
  test si une proposition est dans la conclution d'une des règle de la bdc

4.1.1 Description détaillée

fichier contenant les implémentations des fonctions lié à la structure de donnée Base de Connaissances

Auteur

Florian Cloarec

Version

0.1

Date

21 November 2020

Copyright

GNU GENERAL PUBLIC LICENSE

4.1.2 Documentation des fonctions

4.1.2.1 addHeadBDC()

```
BDConnaissances addHeadBDC (
            BDConnaissances bdc,
            Regle * aAjouter )
```

ajoute une règle à une base de connaissances en tête

Paramètres

bdc	: bdc à laquelle on veut ajouter la règle
aAjouter	: pointeur sur la règle à ajouter

Renvoie

BDConnaissances : renvoie la BDC

Auteur

Florian CLOAREC

Définition à la ligne 19 du fichier BDConnaissances.c.

4.1.2.2 addRegleBDC()

```
BDConnaissances addRegleBDC (
            BDConnaissances bdc,
            Premisse * pListProp,
           char * descriptionPremisse[],
           long nbElemPremisse,
           char * descriptionConclusion )
```

ajoute une règle à une BDC à partir de chaine de caractère

Paramètres

bdc	: base de connaissaint à laquel on veut ajouter la règle
pListProp	: pointeur sur la liste de toute les proposition qui on déjà été ulitilisé
descriptionPremisse	: tableau de chaine de caracètre contenant toutes les descriptions des prémisses à ajouter à la règle
nbElemPremisse	: nombre d'élément que l'on veut mettre dans la prémisse
descriptionConclusion	: desciption à donner à la conlusion

Renvoie

BDConnaissances : renvoie la bdc

Auteur

Florian CLOAREC

Définition à la ligne 69 du fichier BDConnaissances.c.

4.1.2.3 afficheBDC()

```
void afficheBDC (
              BDConnaissances bdc )
```

affiche la totalitée d'une base de connaissances

Paramètres

bdc	: bdc à afficher

Auteur

Florian CLOAREC

Définition à la ligne 60 du fichier BDConnaissances.c.

4.1.2.4 createBDVerite()

```
Premisse createBDVerite (
              Premisse listProp,
              Premisse BDVerite )
```

crée la base de connaissance en y ajoutant toutee les proposition qui sont vraie

Paramètres

listProp	: liste de toute les proposition
BDVerite	: base de vérité que l'on veut créée (mettre à NULL)

Renvoie

Premisse (p. 6) : renvoie la **Premisse** (p. 6) qui à été créée

Auteur

Florian CLOAREC

Définition à la ligne 113 du fichier BDConnaissances.c.

4.1.2.5 deleteAllBDC()

```
void deleteAllBDC (
            BDConnaissances bdc )
```

supprime récursivement la totalitée d'une base de connaissance

Paramètres

bdc	: bdc à supprimer

Auteur

Florian CLOAREC

Définition à la ligne 52 du fichier BDConnaissances.c.

4.1.2.6 deleteHeadBDC()

```
BDConnaissances deleteHeadBDC (
            BDConnaissances bdc )
```

supprime l'élément en tête de la bdc

Paramètres

bdc	: structure dont on veut supprimer l'élément

Renvoie

BDConnaissances : pointeur sur le premier élément de la base de connaissances

Auteur

Florian CLOAREC

Définition à la ligne 36 du fichier BDConnaissances.c.

4.1.2.7 isEmptyBDC()

```
bool isEmptyBDC (
              BDConnaissances bdc )
```

teste si une BDConnaissances est vide

Paramètres

bdc	: base de connaissances que l'on veut tester

Renvoie

true : si la structure est vide
false : si la structure n'est pas vide

Auteur

Florian CLOAREC

Définition à la ligne 14 du fichier BDConnaissances.c.

4.1.2.8 isPropositionInConclusion()

```
bool isPropositionInConclusion (
              BDConnaissances bdc,
              Proposition * prop )
```

test si une proposition est dans la conclution d'une des règle de la bdc

Paramètres

bdc	: base de connassance dans lauqel on veux chercher
prop	: pointeur sur la proposition que l'on veut tester

Renvoie

true : si la proposition à été trouvé dans la conclution d'une regle
false : si elle n'y est pas

Auteur

Florian CLOAREC

Définition à la ligne 127 du fichier BDConnaissances.c.

4.1.2.9 moteurDInference()

```
Premisse moteurDInference (
            Premisse baseVerite,
            BDConnaissances bdc )
```

recherche à partir de la base de véritée et de la base de connaissances les propositions qui sont vraie

Paramètres

baseVerite	: la liste chainé de Porposition qui sont vraie
bdc	: base de connaissance, liste chainé des regles qui permetent de déduire des proposition vraie

Renvoie

Premisse (p. 6) : une liste chainée de propositions qui sont les proposition qui on été trvouée par la fonction

Auteur

Florian CLOAREC

Définition à la ligne 74 du fichier BDConnaissances.c.

4.2 Référence du fichier src/BDConnaissances.h

fichier contenant les prototype des fonctions liée à la base de connaissances ainsi que la définition de la structure elle meme

```
#include "regle.h"
```

Structures de données

— struct **BDConnaissancesElem**

Définitions de type

— typedef struct **BDConnaissancesElem** **BDConnaissancesElem**
— typedef **BDConnaissancesElem** ∗ **BDConnaissances**
pointeur sur le premier élément de la liste chainée

Fonctions

- bool **isEmptyBDC** (**BDConnaissances** bdc)
  teste si une BDConnaissances est vide
- **BDConnaissances addHeadBDC** (**BDConnaissances** bdc, **Regle** ∗aAjouter)
  ajoute une règle à une base de connaissances en tête
- **BDConnaissances deleteHeadBDC** (**BDConnaissances** bdc)
  supprime l'élément en tête de la bdc
- void **deleteAllBDC** (**BDConnaissances** bdc)
  supprime récursivement la totalitée d'une base de connaissance
- void **afficheBDC** (**BDConnaissances** bdc)
  affiche la totalitée d'une base de connaissances
- **BDConnaissances addRegleBDC** (**BDConnaissances** bdc, **Premisse** ∗pListProp, char ∗description↩ Premisse[ ], long nbElemPremisse, char ∗descriptionConclusion)
  ajoute une règle à une BDC à partir de chaine de caractère
- **Premisse moteurDInference** (**Premisse** baseVerite, **BDConnaissances** bdc)
  recherche à partir de la base de véritée et de la base de connaissances les propositions qui sont vraie
- **Premisse createBDVerite** (**Premisse** listProp, **Premisse** BDVerite)
  crée la base de connaissance en y ajoutant toutee les proposition qui sont vraie
- bool **isPropositionInConclusion** (**BDConnaissances** bdc, **Proposition** ∗prop)
  test si une proposition est dans la conclution d'une des règle de la bdc

4.2.1 Description détaillée

fichier contenant les prototype des fonctions liée à la base de connaissances ainsi que la définition de la structure elle meme

Auteur

Florian Cloarec

Version

0.1

Date

21 November 2020

Copyright

GNU GENERAL PUBLIC LICENSE

4.2.2 Documentation des définitions de type

4.2.2.1 BDConnaissances

```
typedef BDConnaissancesElem* BDConnaissances
```

pointeur sur le premier élément de la liste chainée

Auteur

Florian CLOAREC

Définition à la ligne 34 du fichier BDConnaissances.h.

4.2.2.2 BDConnaissancesElem

```
typedef struct  BDConnaissancesElem  BDConnaissancesElem
```

4.2.3 Documentation des fonctions

4.2.3.1 addHeadBDC()

```
BDConnaissances addHeadBDC (
            BDConnaissances bdc,
            Regle * aAjouter )
```

ajoute une règle à une base de connaissances en tête

Paramètres

bdc	: bdc à laquelle on veut ajouter la règle
aAjouter	: pointeur sur la règle à ajouter

Renvoie

BDConnaissances : renvoie la BDC

Auteur

Florian CLOAREC

Définition à la ligne 19 du fichier BDConnaissances.c.

4.2.3.2 addRegleBDC()

```
BDConnaissances addRegleBDC (
            BDConnaissances bdc,
            Premisse * pListProp,
           char * descriptionPremisse[],
           long nbElemPremisse,
           char * descriptionConclusion )
```

ajoute une règle à une BDC à partir de chaine de caractère

Paramètres

bdc	: base de connaissaint à laquel on veut ajouter la règle
pListProp	: pointeur sur la liste de toute les proposition qui on déjà été ulitilisé
descriptionPremisse	: tableau de chaine de caracètre contenant toutes les descriptions des prémisses à ajouter à la règle
nbElemPremisse	: nombre d'élément que l'on veut mettre dans la prémisse
descriptionConclusion	: desciption à donner à la conlusion

Renvoie

BDConnaissances : renvoie la bdc

Auteur

Florian CLOAREC

Définition à la ligne 69 du fichier BDConnaissances.c.

4.2.3.3 afficheBDC()

```
void afficheBDC (
              BDConnaissances bdc )
```

affiche la totalitée d'une base de connaissances

Paramètres

bdc	: bdc à afficher

Auteur

Florian CLOAREC

Définition à la ligne 60 du fichier BDConnaissances.c.

4.2.3.4 createBDVerite()

```
Premisse createBDVerite (
              Premisse listProp,
              Premisse BDVerite )
```

crée la base de connaissance en y ajoutant toutee les proposition qui sont vraie

Paramètres

listProp	: liste de toute les proposition
BDVerite	: base de vérité que l'on veut créée (mettre à NULL)

Renvoie

Premisse (p. 6) : renvoie la **Premisse** (p. 6) qui à été créée

Auteur

Florian CLOAREC

Définition à la ligne 113 du fichier BDConnaissances.c.

4.2.3.5 deleteAllBDC()

```
void deleteAllBDC (
              BDConnaissances bdc )
```

supprime récursivement la totalitée d'une base de connaissance

Paramètres

bdc	: bdc à supprimer

Auteur

Florian CLOAREC

Définition à la ligne 52 du fichier BDConnaissances.c.

4.2.3.6 deleteHeadBDC()

```
BDConnaissances deleteHeadBDC (
              BDConnaissances bdc )
```

supprime l'élément en tête de la bdc

Paramètres

bdc	: structure dont on veut supprimer l'élément

Renvoie

BDConnaissances : pointeur sur le premier élément de la base de connaissances

Auteur

Florian CLOAREC

Définition à la ligne 36 du fichier BDConnaissances.c.

4.2.3.7 isEmptyBDC()

```
bool isEmptyBDC (
              BDConnaissances bdc )
```

teste si une BDConnaissances est vide

Paramètres

bdc	: base de connaissances que l'on veut tester

Renvoie

true : si la structure est vide
false : si la structure n'est pas vide

Auteur

Florian CLOAREC

Définition à la ligne 14 du fichier BDConnaissances.c.

4.2.3.8 isPropositionInConclusion()

```
bool isPropositionInConclusion (
            BDConnaissances bdc,
            Proposition * prop )
```

test si une proposition est dans la conclution d'une des règle de la bdc

Paramètres

bdc	: base de connassance dans lauqel on veux chercher
prop	: pointeur sur la proposition que l'on veut tester

Renvoie

true : si la proposition à été trouvé dans la conclution d'une regle
false : si elle n'y est pas

Auteur

Florian CLOAREC

Définition à la ligne 127 du fichier BDConnaissances.c.

4.2.3.9 moteurDInference()

```
Premisse moteurDInference (
            Premisse baseVerite,
            BDConnaissances bdc )
```

recherche à partir de la base de véritée et de la base de connaissances les propositions qui sont vraie

Paramètres

baseVerite	: la liste chainé de Porposition qui sont vraie
bdc	: base de connaissance, liste chainé des regles qui permetent de déduire des proposition vraie

Renvoie

Premisse (p. 6) : une liste chainée de propositions qui sont les proposition qui on été trvouée par la fonction

Auteur

Florian CLOAREC

Définition à la ligne 74 du fichier BDConnaissances.c.

4.3 Référence du fichier src/fichier.c

fichier contenant les implémentations des fonctions liée à lacréation de la base de connaissances via la lectures de fichier

```
#include "main.h"
```

Fonctions

- **BDConnaissances ReadBDC** (**BDConnaissances** bdc, **Premisse** ∗listeProposition, char chemin↩ Fichier[ ])
  lis le fichier de BDC.csv qui contient toutes les connaissance et l'ajoute à la base de Connaisance
- **BDConnaissances WriteBDC** (**BDConnaissances** bdc, **Premisse** ∗listeProposition, char chemin↩ Fichier[ ])
  écris de nouvelles règles entrées par l'utilisateur dans le fichier de BDC.csv qui contient toutes les connaissance et l'ajoute à la base de Connaisance.

4.3.1 Description détaillée

fichier contenant les implémentations des fonctions liée à lacréation de la base de connaissances via la lectures de fichier

Auteur

Carlo AZANCOTH

Version

0.1

Date

18 décembre 2020

Copyright

GNU GENERAL PUBLIC LICENSE

4.3.2 Documentation des fonctions

4.3.2.1 ReadBDC()

```
BDConnaissances ReadBDC (
            BDConnaissances bdc,
            Premisse * listeProposition,
           char cheminFichier[] )
```

lis le fichier de BDC.csv qui contient toutes les connaissance et l'ajoute à la base de Connaisance

Paramètres

listeProposition	: liste de toute les proposition
bdc	: base de connaissance, liste chainé des regles qui permetent de déduire des proposition vraie
cheminFichier	: chemin vers le ficher qui contient les règle de la bdc

Renvoie

0 si le fichier ne s'est pas ouvert

Auteur

Carlo AZANCOTH

Définition à la ligne 14 du fichier fichier.c.

4.3.2.2 WriteBDC()

```
BDConnaissances WriteBDC (
            BDConnaissances bdc,
            Premisse * listeProposition,
           char cheminFichier[] )
```

écris de nouvelles règles entrées par l'utilisateur dans le fichier de BDC.csv qui contient toutes les connaissance et l'ajoute à la base de Connaisance.

Paramètres

listeProposition	: liste de toute les proposition
bdc	: base de connaissance, liste chainé des regles qui permetent de déduire des proposition vraie
cheminFichier	: chemin vers le ficher qui contient les règle de la bdc

Renvoie

0 si le fichier ne s'est pas ouvert

Auteur

Carlo AZANCOTH

Définition à la ligne 121 du fichier fichier.c.

4.4 Référence du fichier src/fichier.h

fichier contenant les prototype des fonctions liée à lacréation de la base de connaissances via la lectures de fichier

Fonctions

- **BDConnaissances ReadBDC** (**BDConnaissances** bdc, **Premisse** ∗listeProposition, char chemin↩ Fichier[ ])
  lis le fichier de BDC.csv qui contient toutes les connaissance et l'ajoute à la base de Connaisance
- **BDConnaissances WriteBDC** (**BDConnaissances** bdc, **Premisse** ∗listeProposition, char chemin↩ Fichier[ ])
  écris de nouvelles règles entrées par l'utilisateur dans le fichier de BDC.csv qui contient toutes les connaissance et l'ajoute à la base de Connaisance.

4.4.1 Description détaillée

fichier contenant les prototype des fonctions liée à lacréation de la base de connaissances via la lectures de fichier

Auteur

Carlo AZANCOTH

Version

0.1

Date

18 décembre 2020

Copyright

GNU GENERAL PUBLIC LICENSE

4.4.2 Documentation des fonctions

4.4.2.1 ReadBDC()

```
BDConnaissances ReadBDC (
            BDConnaissances bdc,
            Premisse * listeProposition,
           char cheminFichier[] )
```

lis le fichier de BDC.csv qui contient toutes les connaissance et l'ajoute à la base de Connaisance

Paramètres

listeProposition	: liste de toute les proposition
bdc	: base de connaissance, liste chainé des regles qui permetent de déduire des proposition vraie
cheminFichier	: chemin vers le ficher qui contient les règle de la bdc

Renvoie

0 si le fichier ne s'est pas ouvert

Auteur

Carlo AZANCOTH

Définition à la ligne 14 du fichier fichier.c.

4.4.2.2 WriteBDC()

```
BDConnaissances WriteBDC (
             BDConnaissances bdc,
             Premisse * listeProposition,
            char cheminFichier[] )
```

écris de nouvelles règles entrées par l'utilisateur dans le fichier de BDC.csv qui contient toutes les connaissance et l'ajoute à la base de Connaisance.

Paramètres

listeProposition	: liste de toute les proposition
bdc	: base de connaissance, liste chainé des regles qui permetent de déduire des proposition vraie
cheminFichier	: chemin vers le ficher qui contient les règle de la bdc

Renvoie

0 si le fichier ne s'est pas ouvert

Auteur

Carlo AZANCOTH

Définition à la ligne 121 du fichier fichier.c.

4.5 Référence du fichier src/interface.c

fichier contenant les implémentation des fonction lié à l'interface avec l'utilisateur

```
#include "main.h"
```

Fonctions

- void **menuPrincipal** ()
  lance le menu principal du prgramme
- int **acquisitionEntierSecurise** ()
  acquèrt auprès de l'utilisateur un entier de manière sécurisé
- int **acquisitionEntierSansMessageAvecConsigne** (int min, int max, char consigne[ ])
  acquièrt un entier saisi par l'utilisateur avec contrôle d'érreur mais sans message en affichant une consigne
- **Premisse genereBDVerite** (**Premisse** listProp, **Premisse** bdVerite, **BDConnaissances** bdc)
  genere la base de Verite en posant des question à l'utilisateur
- void **systemExpert** (char cheminFicher[ ])
  lance le system expert
- int **verificationPropositionAvecMessage** (char ∗proposition)
  verifie si une proposition entrée par l'utilisateur à été correctement ecrite

4.5.1 Description détaillée

fichier contenant les implémentation des fonction lié à l'interface avec l'utilisateur

Auteur

Florian Cloarec

Version

0.1

Date

07 December 2020

Copyright

GNU GENERAL PUBLIC LICENSE

4.5.2 Documentation des fonctions

4.5.2.1 acquisitionEntierSansMessageAvecConsigne()

```
int acquisitionEntierSansMessageAvecConsigne (
            int min,
            int max,
            char consigne[] )
```

acquièrt un entier saisi par l'utilisateur avec contrôle d'érreur mais sans message en affichant une consigne

Paramètres

min	: valeur minimum que l'utilisateur doit saisir
max	: valeur maximum que l'utilisateur doit saisir
consigne	: consigle à afficher à l'utilsateur

Renvoie

int : revoie la valeur saisi par l'utilsateur et contôlée

Auteur

Florian CLOAREC && Carlo AZANCOTH

Définition à la ligne 128 du fichier interface.c.

4.5.2.2 acquisitionEntierSecurise()

```
int acquisitionEntierSecurise ( )
```

acquèrt auprès de l'utilisateur un entier de manière sécurisé

Renvoie

int : entier saisi par l'utilisateur

Auteur

Florian CLOAREC && Carlo AZANCOTH

Lecture de l'entrée du joueur de type char

Test si le premier caractère est un 0

Conversion du char en int

Définition à la ligne 109 du fichier interface.c.

4.5.2.3 genereBDVerite()

```
Premisse genereBDVerite (
            Premisse listProp,
            Premisse bdVerite,
            BDConnaissances bdc )
```

genere la base de Verite en posant des question à l'utilisateur

Paramètres

listProp	: liste de toute les proposition qui on été enregistré
bdVerite	: liste de toute les propostion qui sont vraie
bdc	: base ce connaissance avec laquel on travail

Renvoie

Premisse (p. 6) : renvoie une liste chainée de proposition qui est la liste de toute les proposition qui sont vraie

Auteur

Florian CLOAREC

Définition à la ligne 138 du fichier interface.c.

4.5.2.4 menuPrincipal()

```
void menuPrincipal ( )
```

lance le menu principal du prgramme

Auteur

Florian CLOAREC

Définition à la ligne 14 du fichier interface.c.

4.5.2.5 systemExpert()

```
void systemExpert (
            char cheminFicher[] )
```

lance le system expert

Paramètres

cheminFicher	: chaine de caracète qui contient le chemin vers le ficher qui permet de générer la bdc

Auteur

Florian CLOAREC

Définition à la ligne 170 du fichier interface.c.

4.5.2.6 verificationPropositionAvecMessage()

```
int verificationPropositionAvecMessage (
            char * proposition )
```

verifie si une proposition entrée par l'utilisateur à été correctement ecrite

Paramètres

proposition	: ponteur vers la proposition à vérifier

Auteur

Carlo AZANCOTH

Définition à la ligne 201 du fichier interface.c.

4.6 Référence du fichier src/interface.h

fichier contenant les prototype des fonction liées à l'interface avec l'utilisateur

Fonctions

- void **menuPrincipal** ()
  lance le menu principal du prgramme
- int **acquisitionEntierSecurise** ()
  acquèrt auprès de l'utilisateur un entier de manière sécurisé
- int **acquisitionEntierSansMessageAvecConsigne** (int min, int max, char consigne[ ])
  acquièrt un entier saisi par l'utilisateur avec contrôle d'érreur mais sans message en affichant une consigne
- **Premisse genereBDVerite** (**Premisse** listProp, **Premisse** bdVerite, **BDConnaissances** bdc)
  genere la base de Verite en posant des question à l'utilisateur
- void **systemExpert** (char cheminFicher[ ])
  lance le system expert
- int **verificationPropositionAvecMessage** (char ∗proposition)
  verifie si une proposition entrée par l'utilisateur à été correctement ecrite

4.6.1 Description détaillée

fichier contenant les prototype des fonction liées à l'interface avec l'utilisateur

Auteur

Florian Cloarec

Version

0.1

Date

07 December 2020

Copyright

GNU GENERAL PUBLIC LICENSE

4.6.2 Documentation des fonctions

4.6.2.1 acquisitionEntierSansMessageAvecConsigne()

```
int acquisitionEntierSansMessageAvecConsigne (
            int min,
            int max,
            char consigne[] )
```

acquièrt un entier saisi par l'utilisateur avec contrôle d'érreur mais sans message en affichant une consigne

Paramètres

min	: valeur minimum que l'utilisateur doit saisir
max	: valeur maximum que l'utilisateur doit saisir
consigne	: consigle à afficher à l'utilsateur

Renvoie

int : revoie la valeur saisi par l'utilsateur et contôlée

Auteur

Florian CLOAREC && Carlo AZANCOTH

Définition à la ligne 128 du fichier interface.c.

4.6.2.2 acquisitionEntierSecurise()

```
int acquisitionEntierSecurise ( )
```

acquèrt auprès de l'utilisateur un entier de manière sécurisé

Renvoie

int : entier saisi par l'utilisateur

Auteur

Florian CLOAREC && Carlo AZANCOTH

Lecture de l'entrée du joueur de type char

Test si le premier caractère est un 0

Conversion du char en int

Définition à la ligne 109 du fichier interface.c.

4.6.2.3 genereBDVerite()

```
Premisse genereBDVerite (
            Premisse listProp,
            Premisse bdVerite,
            BDConnaissances bdc )
```

genere la base de Verite en posant des question à l'utilisateur

Paramètres

listProp	: liste de toute les proposition qui on été enregistré
bdVerite	: liste de toute les propostion qui sont vraie
bdc	: base ce connaissance avec laquel on travail

Renvoie

Premisse (p. 6) : renvoie une liste chainée de proposition qui est la liste de toute les proposition qui sont vraie

Auteur

Florian CLOAREC

Définition à la ligne 138 du fichier interface.c.

4.6.2.4 menuPrincipal()

```
void menuPrincipal ( )
```

lance le menu principal du prgramme

Auteur

Florian CLOAREC

Définition à la ligne 14 du fichier interface.c.

4.6.2.5 systemExpert()

```
void systemExpert (
            char cheminFicher[] )
```

lance le system expert

Paramètres

cheminFicher	: chaine de caracète qui contient le chemin vers le ficher qui permet de générer la bdc

Auteur

Florian CLOAREC

Définition à la ligne 170 du fichier interface.c.

4.6.2.6 verificationPropositionAvecMessage()

```
int verificationPropositionAvecMessage (
            char * proposition )
```

verifie si une proposition entrée par l'utilisateur à été correctement ecrite

Paramètres

proposition	: ponteur vers la proposition à vérifier

Auteur

Carlo AZANCOTH

Définition à la ligne 201 du fichier interface.c.

4.7 Référence du fichier src/main.c

fichier contenant la fonction main

```
#include "main.h"
```

Fonctions

— int **main** (int argc, char ∗argv[ ])
fonction main qui appelle toutes les autres fonction du programme

4.7.1 Description détaillée

fichier contenant la fonction main

Auteur

Florian CLOAREC

Version

0.1

Date

2020-11-14

4.7.2 Documentation des fonctions

4.7.2.1 main()

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

fonction main qui appelle toutes les autres fonction du programme

Paramètres

argc	: nombre d'arguement
argv	: tableu contenant les arguement

Renvoie

int : vaut EXIT_SUCCESS (0) si tout c'est bien passé et EXIT_FAILLURE (1) si il y à eut un problème lors de l'éxécution

Définition à la ligne 19 du fichier main.c.

4.8 Référence du fichier src/main.h

fichier contenant les inclusion de tout les fichier header du projet, à inclure au début de chaque ficher

```
#include "struct.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include "proposition.h"
#include "premisse.h"
#include "regle.h"
#include "BDConnaissances.h"
#include "interface.h"
#include "fichier.h"
```

Macros

— #define **TAILLE_MAXI_PROPOSITION** 100

4.8.1 Description détaillée

fichier contenant les inclusion de tout les fichier header du projet, à inclure au début de chaque ficher

Auteur

Florian Cloarec

Version

0.1

Date

16 November 2020

Copyright

GNU GENERAL PUBLIC LICENSE

4.8.2 Documentation des macros

4.8.2.1 TAILLE_MAXI_PROPOSITION

```
#define TAILLE_MAXI_PROPOSITION 100
```

Définition à la ligne 18 du fichier main.h.

4.9 Référence du fichier src/premisse.c

fichier contenant les implémentation des fonction associé à la structure de donnée **Premisse** (p. 6)

```
#include "main.h"
```

Fonctions

- void **deletePremisse** (**Premisse** prem)
  supprime de la mémoire une prémisse
- **Premisse** **addHeadPremisse** (**Premisse** prem, **Proposition** ∗prop)
  ajouter une proposition en tête d'une prémisse
- **Premisse** **addTailPremisse** (**Premisse** prem, **Proposition** ∗prop)
  ajoute une proposition en queue à la liste chianée de proposition qu'est la prémisse
- void **affichePremisse** (**Premisse** prem)
  affiche une Prémisse de manière récursive
- bool **propositionDansPremisse** (**Premisse** prem, **Proposition** ∗prop)
  teste si une Poposition appartient à une prémisse
- **Premisse** **rechercheSupprimePremisse** (**Premisse** prem, **Proposition** ∗prop)
  supprime une proposition dans la prémisse d'une règle
- bool **premisseIsEmpty** (**Premisse** prem)
  vérifie si une premisse est vide ou ne l'est pas
- **Premisse** **addPropositionUnique** (**Premisse** listProp, char desciption[ ], bool validite)
  ajoute et crée la proposition dans la prémisse si elle n'y est pas déjà
- **Proposition** ∗ **rechercheListProposition** (**Premisse** listProp, char desciption[ ])
  cherche si il existe une proposition avec la meme desciption dans une liste de proposition
- void **deletePremisseProposition** (**Premisse** prem)
  supprime une prémisse et libière de la mémoire les propostion qu'elle contient
- bool **isPremisseTrue** (**Premisse** prem)
  teste si toute les propositon d'une prémisse sont vraie

4.9.1 Description détaillée

fichier contenant les implémentation des fonction associé à la structure de donnée **Premisse** (p. 6)

Auteur

Florian Cloarec

Version

0.1

Date

16 November 2020

Copyright

GNU GENERAL PUBLIC LICENSE

4.9.2 Documentation des fonctions

4.9.2.1 addHeadPremisse()

```
Premisse addHeadPremisse (
            Premisse prem,
            Proposition * prop )
```

ajouter une proposition en tête d'une prémisse

Paramètres

prem	: prémisse dans laquel on veut ajouter
prop	: pointeur sur la proposition à ajouter

Renvoie

Premisse (p. 6) : renvoit la pemisse

Auteur

Florian CLOAREC

Définition à la ligne 24 du fichier premisse.c.

4.9.2.2 addPropositionUnique()

```
Premisse addPropositionUnique (
             Premisse listProp,
            char desciption[],
            bool validite )
```

ajoute et crée la proposition dans la prémisse si elle n'y est pas déjà

Paramètres

listProp	: **Premisse** (p. 6) dans laquel on veux ajouter
desciption	: desciption de la proposition à ajouter
validite	: validite de la proposition à ajouter

Renvoie

Premisse (p. 6) : renvoie la **Premisse** (p. 6) sur laquel on travail

Auteur

Florian CLOAREC

Définition à la ligne 162 du fichier premisse.c.

4.9.2.3 addTailPremisse()

```
Premisse addTailPremisse (
            Premisse prem,
            Proposition * prop )
```

ajoute une proposition en queue à la liste chianée de proposition qu'est la prémisse

Paramètres

prem	: prémisse à qui on veut ajouter une proposition
prop	: proposotion à ajouter

Renvoie

Premisse (p. 6) : renvoit la pemisse

Auteur

Florian CLOAREC

Définition à la ligne 39 du fichier premisse.c.

4.9.2.4 affichePremisse()

```
void affichePremisse (
            Premisse prem )
```

affiche une Prémisse de manière récursive

Paramètres

prem	: prémisse que l'on veut afficher

Auteur

Florian CLOAREC

Définition à la ligne 84 du fichier premisse.c.

4.9.2.5 deletePremisse()

```
void deletePremisse (
              Premisse prem )
```

supprime de la mémoire une prémisse

Paramètres

prem	: premisse que l'on veut supprimer

Auteur

Florian CLOAREC

Définition à la ligne 15 du fichier premisse.c.

4.9.2.6 deletePremisseProposition()

```
void deletePremisseProposition (
              Premisse prem )
```

supprime une prémisse et libière de la mémoire les propostion qu'elle contient

Paramètres

prem	: premisse à suprimmer

Auteur

Florian CLOAREC

Définition à la ligne 222 du fichier premisse.c.

4.9.2.7 isPremisseTrue()

```
bool isPremisseTrue (
              Premisse prem )
```

teste si toute les propositon d'une prémisse sont vraie

Paramètres

prem	: premisse à tester

Renvoie

true : renvoie 1 si toute les proposition de la prémisse sont vrais
false : renvoie 0 sinon

Auteur

Florian CLOAREC

Définition à la ligne 231 du fichier premisse.c.

4.9.2.8 premisseIsEmpty()

```
bool premisseIsEmpty (
            Premisse prem )
```

vérifie si une premisse est vide ou ne l'est pas

Paramètres

Prem	: premisse à vérifier

Renvoie

1 : si la premisse est vide
0 : si la premisse n'est pas vide

Auteur

Carlo AZANCOTH

Définition à la ligne 157 du fichier premisse.c.

4.9.2.9 propositionDansPremisse()

```
bool propositionDansPremisse (
            Premisse prem,
            Proposition * prop )
```

teste si une Poposition appartient à une prémisse

Paramètres

prem	: prémisse à tester
prop	: propostion à rechercher

Renvoie

true : si la propostion à été trouvée dans la premisse
false : si la propostion n'à pas été trouvée dans la premisse

Auteur

Florian CLOAREC

Définition à la ligne 107 du fichier premisse.c.

4.9.2.10 rechercheListProposition()

```
Proposition* rechercheListProposition (
             Premisse listProp,
            char desciption[] )
```

cherche si il existe une proposition avec la meme desciption dans une liste de proposition

Paramètres

listProp	: premisse à tester
desciption	: desciption à rechercher

Renvoie

Premisse (p. 6) : renvoie un pointeur sur la proposition si on trouve et sinon renvoie NULL

Auteur

Florian CLOAREC

Définition à la ligne 206 du fichier premisse.c.

4.9.2.11 rechercheSupprimePremisse()

```
Premisse rechercheSupprimePremisse (
             Premisse prem,
             Proposition * prop )
```

supprime une proposition dans la prémisse d'une règle

Paramètres

prem	: prémisse dont on veut suprimer une proposition
prop	: proposition à supprimer

Renvoie

renvoie la prémisse

Auteur

Florian CLOAREC

Définition à la ligne 123 du fichier premisse.c.

4.10 Référence du fichier src/premisse.h

fichier contenant les prototype des fonctions liées à la structure de données **Premisse** (p. 6) et la définition de la structure **Premisse** (p. 6)

```
#include "proposition.h"
```

Structures de données

— struct **PremisseElem**
structure de donnée permetant de stocker un élément de la liste chainé de Proposion qui représente la Prémisse

Définitions de type

— typedef struct **PremisseElem** **PremisseElem**
— typedef **PremisseElem** ∗ **Premisse**

Fonctions

— void **deletePremisse** (**Premisse** prem)
supprime de la mémoire une prémisse
— **Premisse** **addHeadPremisse** (**Premisse** prem, **Proposition** ∗prop)
ajouter une proposition en tête d'une prémisse
— **Premisse** **addTailPremisse** (**Premisse** prem, **Proposition** ∗prop)
ajoute une proposition en queue à la liste chianée de proposition qu'est la prémisse
— void **affichePremisse** (**Premisse** prem)
affiche une Prémisse de manière récursive
— bool **propositionDansPremisse** (**Premisse** prem, **Proposition** ∗prop)
teste si une Poposition appartient à une prémisse
— **Premisse** **rechercheSupprimePremisse** (**Premisse** prem, **Proposition** ∗prop)
supprime une proposition dans la prémisse d'une règle
— bool **premisseIsEmpty** (**Premisse** prem)
vérifie si une premisse est vide ou ne l'est pas
— **Premisse** **addPropositionUnique** (**Premisse** listProp, char desciption[ ], bool validite)
ajoute et crée la proposition dans la prémisse si elle n'y est pas déjà
— **Proposition** ∗ **rechercheListProposition** (**Premisse** listProp, char desciption[ ])
cherche si il existe une proposition avec la meme desciption dans une liste de proposition
— void **deletePremisseProposition** (**Premisse** prem)
supprime une prémisse et libière de la mémoire les propostion qu'elle contient
— bool **isPremisseTrue** (**Premisse** prem)
teste si toute les propositon d'une prémisse sont vraie

4.10.1 Description détaillée

fichier contenant les prototype des fonctions liées à la structure de données **Premisse** (p. 6) et la définition de la structure **Premisse** (p. 6)

Auteur

Florian Cloarec

Version

0.1

Date

16 November 2020

Copyright

GNU GENERAL PUBLIC LICENSE

4.10.2 Documentation des définitions de type

4.10.2.1 Premisse

```
typedef PremisseElem* Premisse
```

Définition à la ligne 36 du fichier premisse.h.

4.10.2.2 PremisseElem

```
typedef struct PremisseElem PremisseElem
```

4.10.3 Documentation des fonctions

4.10.3.1 addHeadPremisse()

```
Premisse addHeadPremisse (
            Premisse prem,
            Proposition * prop )
```

ajouter une proposition en tête d'une prémisse

Paramètres

prem	: prémisse dans laquel on veut ajouter
prop	: pointeur sur la proposition à ajouter

Renvoie

Premisse (p. 6) : renvoit la pemisse

Auteur

Florian CLOAREC

Définition à la ligne 24 du fichier premisse.c.

4.10.3.2 addPropositionUnique()

```
Premisse addPropositionUnique (
            Premisse listProp,
           char desciption[],
           bool validite )
```

ajoute et crée la proposition dans la prémisse si elle n'y est pas déjà

Paramètres

listProp	: **Premisse** (p. 6) dans laquel on veux ajouter
desciption	: desciption de la proposition à ajouter
validite	: validite de la proposition à ajouter

Renvoie

Premisse (p. 6) : renvoie la **Premisse** (p. 6) sur laquel on travail

Auteur

Florian CLOAREC

Définition à la ligne 162 du fichier premisse.c.

4.10.3.3 addTailPremisse()

```
Premisse addTailPremisse (
            Premisse prem,
            Proposition * prop )
```

ajoute une proposition en queue à la liste chianée de proposition qu'est la prémisse

Paramètres

prem	: prémisse à qui on veut ajouter une proposition
prop	: proposotion à ajouter

Renvoie

Premisse (p. 6) : renvoit la pemisse

Auteur

Florian CLOAREC

Définition à la ligne 39 du fichier premisse.c.

4.10.3.4 affichePremisse()

```
void affichePremisse (
              Premisse prem )
```

affiche une Prémisse de manière récursive

Paramètres

prem	: prémisse que l'on veut afficher

Auteur

Florian CLOAREC

Définition à la ligne 84 du fichier premisse.c.

4.10.3.5 deletePremisse()

```
void deletePremisse (
              Premisse prem )
```

supprime de la mémoire une prémisse

Paramètres

prem	: premisse que l'on veut supprimer

Auteur

Florian CLOAREC

Définition à la ligne 15 du fichier premisse.c.

4.10.3.6 deletePremisseProposition()

```
void deletePremisseProposition (
              Premisse prem )
```

supprime une prémisse et libière de la mémoire les propostion qu'elle contient

Paramètres

prem	: premisse à suprimmer

Auteur

Florian CLOAREC

Définition à la ligne 222 du fichier premisse.c.

4.10.3.7 isPremisseTrue()

```
bool isPremisseTrue (
              Premisse prem )
```

teste si toute les propositon d'une prémisse sont vraie

Paramètres

prem	: premisse à tester

Renvoie

true : renvoie 1 si toute les proposition de la prémisse sont vrais
false : renvoie 0 sinon

Auteur

Florian CLOAREC

Définition à la ligne 231 du fichier premisse.c.

4.10.3.8 premisseIsEmpty()

```
bool premisseIsEmpty (
              Premisse prem )
```

vérifie si une premisse est vide ou ne l'est pas

Paramètres

Prem	: premisse à vérifier

Renvoie

1 : si la premisse est vide
0 : si la premisse n'est pas vide

Auteur

Carlo AZANCOTH

Définition à la ligne 157 du fichier premisse.c.

4.10.3.9 propositionDansPremisse()

```
bool propositionDansPremisse (
              Premisse prem,
              Proposition * prop )
```

teste si une Poposition appartient à une prémisse

Paramètres

prem	: prémisse à tester
prop	: propostion à rechercher

Renvoie

true : si la propostion à été trouvée dans la premisse
false : si la propostion n'à pas été trouvée dans la premisse

Auteur

Florian CLOAREC

Définition à la ligne 107 du fichier premisse.c.

4.10.3.10 rechercheListProposition()

```
Proposition* rechercheListProposition (
            Premisse listProp,
           char desciption[] )
```

cherche si il existe une proposition avec la meme desciption dans une liste de proposition

Paramètres

listProp	: premisse à tester
desciption	: desciption à rechercher

Renvoie

Premisse (p. 6) : renvoie un pointeur sur la proposition si on trouve et sinon renvoie NULL

Auteur

Florian CLOAREC

Définition à la ligne 206 du fichier premisse.c.

4.10.3.11 rechercheSupprimePremisse()

```
Premisse rechercheSupprimePremisse (
            Premisse prem,
            Proposition * prop )
```

supprime une proposition dans la prémisse d'une règle

Paramètres

prem	: prémisse dont on veut suprimer une proposition
prop	: proposition à supprimer

Renvoie

renvoie la prémisse

Auteur

Florian CLOAREC

Définition à la ligne 123 du fichier premisse.c.

4.11 Référence du fichier src/proposition.c

fichier contenant les implémentation des fonctions associée à la structure de donnée Propositon

```
#include "main.h"
```

Fonctions

— **Proposition** ∗ **newProposition** (char valueProposition[ ], bool validite)
*structure de donnée **Proposition** (p. 7) qui est une chaine de caractère*
— void **deleteProposition** (**Proposition** ∗propositionToDelete)
suprime de la mémoire une variable de type proposition
— void **affichePropositon** (**Proposition** ∗proposition)
afficher le contenue d'une propositon
— void **setValidite** (**Proposition** ∗prop, bool val)
définit la valeur de la sous variable validité d'une proposition

4.11.1 Description détaillée

fichier contenant les implémentation des fonctions associée à la structure de donnée Propositon

Auteur

Florian Cloarec

Version

0.1

Date

16 November 2020

Copyright

GNU GENERAL PUBLIC LICENSE

4.11.2 Documentation des fonctions

4.11.2.1 affichePropositon()

```
void affichePropositon (
            Proposition * proposition )
```

afficher le contenue d'une propositon

Paramètres

prposition	: pointeur sur la proposition à afficher

Auteur

Florian CLOAREC

Définition à la ligne 44 du fichier proposition.c.

4.11.2.2 deleteProposition()

```
void deleteProposition (
              Proposition * propositionToDelete )
```

suprime de la mémoire une variable de type proposition

Paramètres

propositionToDelete	: pointeur sur la proposition à supprimer

Auteur

Florian CLOAREC

Définition à la ligne 38 du fichier proposition.c.

4.11.2.3 newProposition()

```
Proposition* newProposition (
            char valueProposition[],
            bool validite )
```

structure de donnée **Proposition** (p. 7) qui est une chaine de caractère

Auteur

Florian CLOAREC

crée une variable de type proposiont

Paramètres

valuePropositon[ ]	: chaine de caractère qui contient la valuer de la propositon à créer
validite	: 1 si la propositon est vraie, 0 si elle est fausse

Renvoie

Proposition (p. 7) : pointeru sur la propositon qui viens d'être crée

Auteur

Florian CLOAREC

Définition à la ligne 14 du fichier proposition.c.

4.11.2.4 setValidite()

```
void setValidite (
              Proposition * prop,
             bool val )
```

définit la valeur de la sous variable validité d'une proposition

Paramètres

prop	: proposition que l'on veut modifier
val	: valeur à donner à validite

Auteur

Florian CLOAREC

Définition à la ligne 49 du fichier proposition.c.

4.12 Référence du fichier src/proposition.h

fichier header contenant les prototypes des fonctions associer à la structure de donnée proposition ainsi que la déclarion de la structure de donnée proposition elle même

Structures de données

— struct **Proposition**
structure de donnée permetant de stocker ou une Propositon

Définitions de type

— typedef struct **Proposition** **Proposition**

Fonctions

— **Proposition** * **newProposition** (char valueProposition[ ], bool validite)
*structure de donnée **Proposition** (p. 7) qui est une chaine de caractère*
— void **deleteProposition** (**Proposition** *propositionToDelete)
suprime de la mémoire une variable de type proposition
— void **affichePropositon** (**Proposition** *proposition)
afficher le contenue d'une propositon
— void **setValidite** (**Proposition** *prop, bool val)
définit la valeur de la sous variable validité d'une proposition

4.12.1 Description détaillée

fichier header contenant les prototypes des fonctions associer à la structure de donnée proposition ainsi que la déclarion de la structure de donnée proposition elle même

Auteur

Florian Cloarec

Version

0.1

Date

16 November 2020

Copyright

GNU GENERAL PUBLIC LICENSE

4.12.2 Documentation des définitions de type

4.12.2.1 Proposition

```
typedef struct  Proposition  Proposition
```

4.12.3 Documentation des fonctions

4.12.3.1 afficheProposiston()

```
void affichePropositon (
            Proposition * proposition )
```

afficher le contenue d'une propositon

Paramètres

prposition	: pointeur sur la proposition à afficher

Auteur

Florian CLOAREC

Définition à la ligne 44 du fichier proposition.c.

4.12.3.2 deleteProposition()

```
void deleteProposition (
              Proposition * propositionToDelete )
```

suprime de la mémoire une variable de type proposition

Paramètres

propositionToDelete	: pointeur sur la proposition à supprimer

Auteur

Florian CLOAREC

Définition à la ligne 38 du fichier proposition.c.

4.12.3.3 newProposition()

```
Proposition* newProposition (
             char valueProposition[],
             bool validite )
```

structure de donnée **Proposition** (p. 7) qui est une chaine de caractère

Auteur

Florian CLOAREC

crée une variable de type proposiont

Paramètres

valuePropositon[ ]	: chaine de caractère qui contient la valuer de la propositon à créer
validite	: 1 si la propositon est vraie, 0 si elle est fausse

Renvoie

Proposition (p. 7) : pointeru sur la propositon qui viens d'être crée

Auteur

Florian CLOAREC

Définition à la ligne 14 du fichier proposition.c.

4.12.3.4 setValidite()

```
void setValidite (
              Proposition * prop,
             bool val )
```

définit la valeur de la sous variable validité d'une proposition

Paramètres

prop	: proposition que l'on veut modifier
val	: valeur à donner à validite

Auteur

Florian CLOAREC

Définition à la ligne 49 du fichier proposition.c.

4.13 Référence du fichier src/regle.c

fichier contenant les implémentation des fonctions relative à la structure de donnée **Regle** (p. 8)

```
#include "main.h"
```

Fonctions

- **Regle** * **newRegle** ()
  crée une règle vide
- void **deleteRegle** (**Regle** *regleToDelete)
  suprime une règle ainsi que tout ces composant
- **Premisse** **addConclusion** (**Regle** *regle, **Premisse** listProp, char *description)
  défini une popotion comme la conclution d'une règle
- void **afficheRegle** (**Regle** *regle)
  affiche une règle
- bool **ReglePremisseIsEmpty** (**Regle** *regleAVerif)
  vérifie si la premisse d'une règle est vide ou ne l'est pas
- **Regle** * **supprimePropositionPremisseRegle** (**Regle** *regle, **Proposition** *prop)
  supprime la proposition de la prémisse d'une règle
- **Proposition** * **conlusionRegle** (**Regle** *regle)
  revoie la conclution d'une règle
- **Premisse** **instertHeadPremisseRegle** (**Regle** *regle, **Premisse** listProp, char *description)
  ajoute en tete une propostion à la prémisse d'une règle
- **Regle** * **createRegle** (**Premisse** *pListProp, char *descriptionPremisse[ ], long nbElemPremisse, char *descriptionConclusion)
  crée une règle à partir de chaine de caractère

4.13.1 Description détaillée

fichier contenant les implémentation des fonctions relative à la structure de donnée **Regle** (p. 8)

Auteur

Florian Cloarec

Version

0.1

Date

14 November 2020

Copyright

GNU GENERAL PUBLIC LICENSE

4.13.2 Documentation des fonctions

4.13.2.1 addConclusion()

```
Premisse addConclusion (
            Regle * regle,
            Premisse listProp,
           char * description )
```

défini une popotion comme la conclution d'une règle

Paramètres

regle	: regle pour laquel on définit la conclution
listProp	: liste de toute les proposition
description	: destription que l'on veut donner à la conclusion

Renvoie

revoie listProp au cas ou il ait changé

Auteur

Florian CLOAREC && Carlo AZANCOTH

Définition à la ligne 34 du fichier regle.c.

4.13.2.2 afficheRegle()

```
void afficheRegle (
              Regle * regle )
```

affiche une règle

Paramètres

regle	: pointeur vers la régle à afficher

Auteur

Carlo AZANCOTH

Définition à la ligne 45 du fichier regle.c.

4.13.2.3 conlusionRegle()

```
Proposition* conlusionRegle (
              Regle * regle )
```

revoie la conclution d'une règle

Paramètres

regle	: règle dont on veut obtenir la conclusion

Renvoie

Proposition (p. 7)

Auteur

Florian CLOAREC

Définition à la ligne 72 du fichier regle.c.

4.13.2.4 createRegle()

```
Regle* createRegle (
             Premisse * pListProp,
            char * descriptionPremisse[],
            long nbElemPremisse,
            char * descriptionConclusion )
```

crée une règle à partir de chaine de caractère

Paramètres

pListProp	: pointeur sur la liste de toute les proposition qui on déjà été ulitilisé
descriptionPremisse	: tableau de chaine de caracètre contenant toutes les descriptions des prémisses à ajouter à la règle
nbElemPremisse	: nombre d'élément que l'on veut mettre dans la prémisse
descriptionConclusion	: desciption à donner à la conlusion

Renvoie

Regle∗ : renvoie un pointeur sur la règle qui à été crée

Auteur

Florian CLOAREC

Définition à la ligne 91 du fichier regle.c.

4.13.2.5 deleteRegle()

```
void deleteRegle (
              Regle * regleToDelete )
```

suprime une règle ainsi que tout ces composant

Paramètres

regleToDelete	: regle que l'on veut suprimer

Auteur

Florian CLOAREC

Définition à la ligne 28 du fichier regle.c.

4.13.2.6 instertHeadPremisseRegle()

```
Premisse instertHeadPremisseRegle (
              Regle * regle,
              Premisse listProp,
             char * description )
```

ajoute en tete une propostion à la prémisse d'une règle

Paramètres

regle	: règle à laquelle on veux ajouter une proposition
listProp	: liste de toute les propositions
description	: description de la proposition à ajouter

Renvoie

revoie listProp au cas ou il ait changé

Auteur

Florian CLOAREC

Définition à la ligne 77 du fichier regle.c.

4.13.2.7 newRegle()

```
Regle* newRegle ( )
```

crée une règle vide

Définition des prototypes des fonctions lié à une Règle

Renvoie

Regle∗

Auteur

Florian CLOAREC

Définition à la ligne 14 du fichier regle.c.

4.13.2.8 ReglePremisseIsEmpty()

```
bool ReglePremisseIsEmpty (
            Regle * regleAVerif )
```

vérifie si la premisse d'une règle est vide ou ne l'est pas

Paramètres

regleAVerif	: regle ou se trouve la premisse à vérifier

Renvoie

1 : si la premisse est vide
0 : si la premisse n'est pas vide

Auteur

Carlo AZANCOTH

Définition à la ligne 61 du fichier regle.c.

4.13.2.9 supprimePropositionPremisseRegle()

```
Regle* supprimePropositionPremisseRegle (
            Regle * regle,
            Proposition * prop )
```

supprime la proposition de la prémisse d'une règle

Paramètres

regle	: règle dans laquele on veut supprimer
prop	: propostion à rechercher et supprimer

Renvoie

pointeur sur la règle avec laquel on à travaillé

Auteur

Florian CLOAREC

Définition à la ligne 65 du fichier regle.c.

4.14 Référence du fichier src/regle.h

fichier contenant les protoype lier à la structure de donnée **Regle** (p. 8) ainsi que les déclaration des structure de donnée **Regle** (p. 8) et ces composée

```
#include "proposition.h"
#include "premisse.h"
```

Structures de données

— struct **Regle**
structure de donnée qui représente une règle

Définitions de type

— typedef struct **Regle** **Regle**

Fonctions

- **Regle** ∗ **newRegle** ()
  crée une règle vide
- void **deleteRegle** (**Regle** ∗regleToDelete)
  suprime une règle ainsi que tout ces composant
- **Premisse addConclusion** (**Regle** ∗regle, **Premisse** listProp, char ∗description)
  défini une popotion comme la conclution d'une règle
- void **afficheRegle** (**Regle** ∗regle)
  affiche une règle
- bool **ReglePremisseIsEmpty** (**Regle** ∗regleAVerif)
  vérifie si la premisse d'une règle est vide ou ne l'est pas
- **Regle** ∗ **supprimePropositionPremisseRegle** (**Regle** ∗regle, **Proposition** ∗prop)
  supprime la proposition de la prémisse d'une règle
- **Proposition** ∗ **conlusionRegle** (**Regle** ∗regle)
  revoie la conclution d'une règle
- **Premisse instertHeadPremisseRegle** (**Regle** ∗regle, **Premisse** listProp, char ∗description)
  ajoute en tete une propostion à la prémisse d'une règle
- **Regle** ∗ **createRegle** (**Premisse** ∗pListProp, char ∗descriptionPremisse[ ], long nbElemPremisse, char ∗descriptionConclusion)
  crée une règle à partir de chaine de caractère

4.14.1 Description détaillée

fichier contenant les protoype lier à la structure de donnée **Regle** (p. 8) ainsi que les déclaration des structure de donnée **Regle** (p. 8) et ces composée

Auteur

Florian Cloarec

Version

0.1

Date

14 November 2020

Copyright

GNU GENERAL PUBLIC LICENSE

4.14.2 Documentation des définitions de type

4.14.2.1 Regle

```
typedef struct Regle Regle
```

4.14.3 Documentation des fonctions

4.14.3.1 addConclusion()

```
Premisse addConclusion (
            Regle * regle,
            Premisse listProp,
           char * description )
```

défini une popotion comme la conclution d'une règle

Paramètres

regle	: regle pour laquel on définit la conclution
listProp	: liste de toute les proposition
description	: destription que l'on veut donner à la conclusion

Renvoie

revoie listProp au cas ou il ait changé

Auteur

Florian CLOAREC && Carlo AZANCOTH

Définition à la ligne 34 du fichier regle.c.

4.14.3.2 afficheRegle()

```
void afficheRegle (
              Regle * regle )
```

affiche une règle

Paramètres

regle	: pointeur vers la régle à afficher

Auteur

Carlo AZANCOTH

Définition à la ligne 45 du fichier regle.c.

4.14.3.3 conlusionRegle()

```
Proposition* conlusionRegle (
              Regle * regle )
```

revoie la conclution d'une règle

Paramètres

regle	: règle dont on veut obtenir la conclusion

Renvoie

Proposition (p. 7)

Auteur

Florian CLOAREC

Définition à la ligne 72 du fichier regle.c.

4.14.3.4 createRegle()

```
Regle* createRegle (
            Premisse * pListProp,
           char * descriptionPremisse[],
           long nbElemPremisse,
           char * descriptionConclusion )
```

crée une règle à partir de chaine de caractère

Paramètres

pListProp	: pointeur sur la liste de toute les proposition qui on déjà été ulitilisé
descriptionPremisse	: tableau de chaine de caracètre contenant toutes les descriptions des prémisses à ajouter à la règle
nbElemPremisse	: nombre d'élément que l'on veut mettre dans la prémisse
descriptionConclusion	: desciption à donner à la conlusion

Renvoie

Regle* : renvoie un pointeur sur la règle qui à été crée

Auteur

Florian CLOAREC

Définition à la ligne 91 du fichier regle.c.

4.14.3.5 deleteRegle()

```
void deleteRegle (
             Regle * regleToDelete )
```

suprime une règle ainsi que tout ces composant

Paramètres

regleToDelete	: regle que l'on veut suprimer

Auteur

Florian CLOAREC

Définition à la ligne 28 du fichier regle.c.

4.14.3.6 instertHeadPremisseRegle()

```
Premisse instertHeadPremisseRegle (
              Regle * regle,
              Premisse listProp,
             char * description )
```

ajoute en tete une propostion à la prémisse d'une règle

Paramètres

regle	: règle à laquelle on veux ajouter une proposition
listProp	: liste de toute les propositions
description	: description de la proposition à ajouter

Renvoie

revoie listProp au cas ou il ait changé

Auteur

Florian CLOAREC

Définition à la ligne 77 du fichier regle.c.

4.14.3.7 newRegle()

```
Regle* newRegle ( )
```

crée une règle vide

Définition des prototypes des fonctions lié à une Règle

Renvoie

Regle∗

Auteur

Florian CLOAREC

Définition à la ligne 14 du fichier regle.c.

4.14.3.8 ReglePremisseIsEmpty()

```
bool ReglePremisseIsEmpty (
              Regle * regleAVerif )
```

vérifie si la premisse d'une règle est vide ou ne l'est pas

Paramètres

regleAVerif	: regle ou se trouve la premisse à vérifier

Renvoie

1 : si la premisse est vide
0 : si la premisse n'est pas vide

Auteur

Carlo AZANCOTH

Définition à la ligne 61 du fichier regle.c.

4.14.3.9 supprimePropositionPremisseRegle()

```
Regle* supprimePropositionPremisseRegle (
            Regle * regle,
            Proposition * prop )
```

supprime la proposition de la prémisse d'une règle

Paramètres

regle	: règle dans laquele on veut supprimer
prop	: propostion à rechercher et supprimer

Renvoie

pointeur sur la règle avec laquel on à travaillé

Auteur

Florian CLOAREC

Définition à la ligne 65 du fichier regle.c.

4.15 Référence du fichier src/struct.h

fichier contenant les structures de données et les énumérations

4.15.1 Description détaillée

fichier contenant les structures de données et les énumérations

Auteur

Florian Cloarec

Version

0.1

Date

14 November 2020

Copyright

Index