

Projet_LO21

1.0

Généré par Doxygen 1.8.18

1 Index des structures de données	1
1.1 Structures de données	1
2 Index des fichiers	3
2.1 Liste des fichiers	3
3 Documentation des structures de données	5
3.1 Référence de la structure BDConnaissancesElem	5
3.1.1 Description détaillée	5
3.1.2 Documentation des champs	5
3.1.2.1 suivant	5
3.1.2.2 valeur	5
3.2 Référence de la structure BDConnaissanceselem	6
3.2.1 Description détaillée	6
3.3 Référence de la structure Premisse	6
3.3.1 Description détaillée	6
3.4 Référence de la structure PremisseElem	6
3.4.1 Description détaillée	7
3.4.2 Documentation des champs	7
3.4.2.1 elemSuivant	7
3.4.2.2 valeur	7
3.5 Référence de la structure Proposition	7
3.5.1 Description détaillée	8
3.5.2 Documentation des champs	8
3.5.2.1 description	8
3.5.2.2 valideite	8
3.6 Référence de la structure Regle	8
3.6.1 Description détaillée	8
3.6.2 Documentation des champs	9
3.6.2.1 conclusion	9
3.6.2.2 premisses	9
4 Documentation des fichiers	11
4.1 Référence du fichier src/BDConnaissances.c	11
4.1.1 Description détaillée	11
4.1.2 Documentation des fonctions	12
4.1.2.1 addHeadBDC()	12
4.1.2.2 addRegleBDC()	12
4.1.2.3 afficheBDC()	13
4.1.2.4 createBDVerite()	13
4.1.2.5 deleteAllBDC()	14
4.1.2.6 deleteHeadBDC()	14
4.1.2.7 isEmptyBDC()	15

4.1.2.8 isPropositionInConclusion()	15
4.1.2.9 moteurDInference()	16
4.2 Référence du fichier src/BDConnaissances.h	16
4.2.1 Description détaillée	17
4.2.2 Documentation des définitions de type	17
4.2.2.1 BDConnaissances	17
4.2.2.2 BDConnaissancesElem	18
4.2.3 Documentation des fonctions	18
4.2.3.1 addHeadBDC()	18
4.2.3.2 addRegleBDC()	18
4.2.3.3 afficheBDC()	19
4.2.3.4 createBDVerite()	19
4.2.3.5 deleteAllBDC()	20
4.2.3.6 deleteHeadBDC()	20
4.2.3.7 isEmptyBDC()	20
4.2.3.8 isPropositionInConclusion()	21
4.2.3.9 moteurDInference()	21
4.3 Référence du fichier src/fichier.c	22
4.3.1 Description détaillée	22
4.3.2 Documentation des fonctions	23
4.3.2.1 ReadBDC()	23
4.3.2.2 WriteBDC()	23
4.4 Référence du fichier src/fichier.h	24
4.4.1 Description détaillée	24
4.4.2 Documentation des fonctions	24
4.4.2.1 ReadBDC()	24
4.4.2.2 WriteBDC()	25
4.5 Référence du fichier src/interface.c	25
4.5.1 Description détaillée	26
4.5.2 Documentation des fonctions	26
4.5.2.1 acquisitionEntierSansMessageAvecConsigne()	26
4.5.2.2 acquisitionEntierSecurise()	27
4.5.2.3 genereBDVerite()	27
4.5.2.4 menuPrincipal()	28
4.5.2.5 systemExpert()	28
4.5.2.6 verificationPropositionAvecMessage()	28
4.6 Référence du fichier src/interface.h	29
4.6.1 Description détaillée	29
4.6.2 Documentation des fonctions	29
4.6.2.1 acquisitionEntierSansMessageAvecConsigne()	29
4.6.2.2 acquisitionEntierSecurise()	30
4.6.2.3 genereBDVerite()	30

4.6.2.4 menuPrincipal()	31
4.6.2.5 systemExpert()	31
4.6.2.6 verificationPropositionAvecMessage()	32
4.7 Référence du fichier src/main.c	32
4.7.1 Description détaillée	32
4.7.2 Documentation des fonctions	32
4.7.2.1 main()	32
4.8 Référence du fichier src/main.h	33
4.8.1 Description détaillée	33
4.8.2 Documentation des macros	34
4.8.2.1 TAILLE_MAXI_PROPOSITION	34
4.9 Référence du fichier src/premise.c	34
4.9.1 Description détaillée	34
4.9.2 Documentation des fonctions	35
4.9.2.1 addHeadPremisse()	35
4.9.2.2 addPropositionUnique()	35
4.9.2.3 addTailPremisse()	36
4.9.2.4 affichePremisse()	36
4.9.2.5 deletePremisse()	37
4.9.2.6 deletePremisseProposition()	37
4.9.2.7 isPremisseTrue()	37
4.9.2.8 premisselsEmpty()	38
4.9.2.9 propositionDansPremisse()	38
4.9.2.10 rechercheListProposition()	39
4.9.2.11 rechercheSupprimePremisse()	39
4.10 Référence du fichier src/premise.h	40
4.10.1 Description détaillée	41
4.10.2 Documentation des définitions de type	41
4.10.2.1 Premisse	41
4.10.2.2 PremisseElem	41
4.10.3 Documentation des fonctions	41
4.10.3.1 addHeadPremisse()	41
4.10.3.2 addPropositionUnique()	42
4.10.3.3 addTailPremisse()	42
4.10.3.4 affichePremisse()	43
4.10.3.5 deletePremisse()	43
4.10.3.6 deletePremisseProposition()	44
4.10.3.7 isPremisseTrue()	44
4.10.3.8 premisselsEmpty()	45
4.10.3.9 propositionDansPremisse()	45
4.10.3.10 rechercheListProposition()	46
4.10.3.11 rechercheSupprimePremisse()	46

4.11	Référence du fichier src/proposition.c	46
4.11.1	Description détaillée	47
4.11.2	Documentation des fonctions	47
4.11.2.1	affichePropositon()	47
4.11.2.2	deleteProposition()	48
4.11.2.3	newProposition()	48
4.11.2.4	setValidite()	49
4.12	Référence du fichier src/proposition.h	49
4.12.1	Description détaillée	50
4.12.2	Documentation des définitions de type	50
4.12.2.1	Proposition	50
4.12.3	Documentation des fonctions	50
4.12.3.1	affichePropositon()	50
4.12.3.2	deleteProposition()	51
4.12.3.3	newProposition()	51
4.12.3.4	setValidite()	52
4.13	Référence du fichier src/regle.c	52
4.13.1	Description détaillée	53
4.13.2	Documentation des fonctions	53
4.13.2.1	addConclusion()	53
4.13.2.2	afficheRegle()	54
4.13.2.3	conclusionRegle()	54
4.13.2.4	createRegle()	54
4.13.2.5	deleteRegle()	55
4.13.2.6	instertHeadPremisseRegle()	55
4.13.2.7	newRegle()	56
4.13.2.8	ReglePremisselsEmpty()	56
4.13.2.9	supprimePropositionPremisseRegle()	57
4.14	Référence du fichier src/regle.h	57
4.14.1	Description détaillée	58
4.14.2	Documentation des définitions de type	58
4.14.2.1	Regle	58
4.14.3	Documentation des fonctions	58
4.14.3.1	addConclusion()	59
4.14.3.2	afficheRegle()	60
4.14.3.3	conclusionRegle()	60
4.14.3.4	createRegle()	61
4.14.3.5	deleteRegle()	61
4.14.3.6	instertHeadPremisseRegle()	62
4.14.3.7	newRegle()	62
4.14.3.8	ReglePremisselsEmpty()	62
4.14.3.9	supprimePropositionPremisseRegle()	63

4.15 Référence du fichier src/struct.h	63
4.15.1 Description détaillée	64
Index	65

Chapitre 1

Index des structures de données

1.1 Structures de données

Liste des structures de données avec une brève description :

BDConnaissancesElem	5
BDConnaissanceselem	
Un élément de la liste chaîné de Règles	6
Premisse	
Strucutre de donné permetant de représenter une prémisse par une liste chaîné de propositon	6
PremisseElem	
Structure de donnée permetant de stocker un élément de la liste chaîné de Proposition qui représente la Prémisse	6
Proposition	
Structure de donnée permetant de stocker ou une Proposition	7
Regle	
Structure de donnée qui représente une règle	8

Chapitre 2

Index des fichiers

2.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

src/ BDConnaissances.c	Fichier contenant les implémentations des fonctions lié à la structure de donnée Base de Connaissances	11
src/ BDConnaissances.h	Fichier contenant les prototype des fonctions liée à la base de connaissances ainsi que la définition de la structure elle meme	16
src/ fichier.c	Fichier contenant les implémentations des fonctions liée à la création de la base de connaissances via la lectures de fichier	22
src/ fichier.h	Fichier contenant les prototype des fonctions liée à la création de la base de connaissances via la lectures de fichier	24
src/ interface.c	Fichier contenant les implémentations des fonction lié à l'interface avec l'utilisateur	25
src/ interface.h	Fichier contenant les prototype des fonction liées à l'interface avec l'utilisateur	29
src/ main.c	Fichier contenant la fonction main	32
src/ main.h	Fichier contenant les inclusion de tout les fichier header du projet, à inclure au début de chaque fichier	33
src/ premise.c	Fichier contenant les implémentations des fonction associé à la structure de donnée Premisse (p. 6)	34
src/ premise.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)	40
src/ proposition.c	Fichier contenant les implémentations des fonctions associée à la structure de donnée Proposition	46
src/ proposition.h	Fichier header contenant les prototypes des fonctions associées à la structure de donnée proposition ainsi que la déclaration de la structure de donnée proposition elle même	49
src/ regle.c	Fichier contenant les implémentations des fonctions relative à la structure de donnée Regle (p. 8)	52

src/ **regle.h**

Fichier contenant les prototype lier à la structure de donnée **Regle** (p. 8) ainsi que les déclaration des structure de donnée **Regle** (p. 8) et ces composée 57

src/ **struct.h**

Fichier contenant les structures de données et les énumérations 63

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 chaînée 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 chaîné de Règles

```
#include <BDConnaissances.h>
```

3.2.1 Description détaillée

un élément de la liste chaîné 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 prémissse par une liste chaîné de propositon

```
#include <premise.h>
```

3.3.1 Description détaillée

strucutre de donné permetant de représenter une prémissse par une liste chaîné de propositon

Auteur

Florian CLOAREC

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

— src/ **premise.h**

3.4 Référence de la structure PremisseElem

structure de donnée permetant de stocker un élément de la liste chaîné de Proposition qui représente la Prémissse

```
#include <premise.h>
```

Champs de données

- **Proposition** * valeur
- struct **PremisseElem** * elemSuivant

3.4.1 Description détaillée

structure de donnée permettant de stocker un élément de la liste chaîné de Proposition qui représente la Prémisse

Auteur

Florian CLOAREC

Définition à la ligne 23 du fichier premisses.h.

3.4.2 Documentation des champs

3.4.2.1 elemSuivant

```
struct PremisseElem* elemSuivant
```

valeur de l'élément de la liste chaîné

Définition à la ligne 26 du fichier premisses.h.

3.4.2.2 valeur

```
Proposition* valeur
```

Définition à la ligne 25 du fichier premisses.h.

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

— src/ **premisses.h**

3.5 Référence de la structure Proposition

structure de donnée permettant de stocker une Proposition

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

Champs de données

— char * **description**
— bool **validite**

3.5.1 Description détaillée

structure de donnée permettant de stocker ou une Proposition

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 valide
```

chaîne de caractère qui contient la description en langage 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** **premise**
- **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 chaîné 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 premisses

Premisse premisses

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 totalité d'une base de connaissance
- **void afficheBDC (BDConnaissances bdc)**
affiche la totalité 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 chaîne de caractère
- **Premisse moteurDInference (Premisse baseVerite, BDConnaissances bdc)**
recherche à partir de la base de vérité 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 toutes les propositions qui sont vraie
- **bool isPropositionInConclusion (BDConnaissances bdc, Proposition *prop)**
test si une proposition est dans la conclusion d'une des règles 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

<i>bdc</i>	: bdc à laquelle on veut ajouter la règle
<i>aAjouter</i>	: 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,
    Premsisse * pListProp,
    char * descriptionPremsisse[],
    long nbElemPremsisse,
    char * descriptionConclusion )
```

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

Paramètres

<i>bdc</i>	: base de connaissance à laquelle on veut ajouter la règle
<i>pListProp</i>	: pointeur sur la liste de toutes les propositions qui ont déjà été utilisées
<i>descriptionPremisse</i>	: tableau de chaîne de caractères contenant toutes les descriptions des prémisses à ajouter à la règle
<i>nbElemPremisse</i>	: nombre d'éléments que l'on veut mettre dans la prémisse
<i>descriptionConclusion</i>	: description à donner à la conclusion

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 totalité d'une base de connaissances

Paramètres

<i>bdc</i>	: 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 toutes les propositions qui sont vraies

Paramètres

<i>listProp</i>	: liste de toutes les propositions
<i>BDVerite</i>	: base de vérité que l'on veut créer (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 totalité d'une base de connaissance

Paramètres

<i>bdc</i>	: 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

<i>bdc</i>	: 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

<i>bdc</i>	: 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 conclusion d'une des règle de la bdc

Paramètres

<i>bdc</i>	: base de connaissance dans laquelle on veut chercher
<i>prop</i>	: pointeur sur la proposition que l'on veut tester

Renvoie

true : si la proposition a été trouvée dans la conclusion d'une règle

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 vérité et de la base de connaissances les propositions qui sont vraies

Paramètres

<i>baseVerite</i>	: la liste chaînée de Proposition qui sont vraies
<i>bdc</i>	: base de connaissance, liste chaînée des règles qui permettent de déduire des propositions vraies

Renvoie

Premisse (p. 6) : une liste chaînée de propositions qui sont les propositions qui ont été trouvées 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 prototypes des fonctions liées à la base de connaissances ainsi que la définition de la structure elle-même

```
#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 chaînée
```


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 totalité d'une base de connaissance
- `void afficheBDC (BDConnaissances bdc)`
affiche la totalité 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 chaîne de caractère
- `Premisse moteurDInference (Premisse baseVerite, BDConnaissances bdc)`
recherche à partir de la base de vérité 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 toutes les propositions qui sont vraie
- `bool isPropositionInConclusion (BDConnaissances bdc, Proposition *prop)`
test si une proposition est dans la conclusion d'une des règles de la bdc

4.2.1 Description détaillée

fichier contenant les prototypes des fonctions liées à la base de connaissances ainsi que la définition de la structure elle-même

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 chaînée

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

<i>bdc</i>	: bdc à laquelle on veut ajouter la règle
<i>aAjouter</i>	: 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 chaîne de caractère

Paramètres

<i>bdc</i>	: base de connaissance à laquelle on veut ajouter la règle
<i>pListProp</i>	: pointeur sur la liste de toutes les propositions qui ont déjà été utilisées
<i>descriptionPremisse</i>	: tableau de chaîne de caractère contenant toutes les descriptions des prémisses à ajouter à la règle
<i>nbElemPremisse</i>	: nombre d'éléments que l'on veut mettre dans la prémisses
<i>descriptionConclusion</i>	: description à donner à la conclusion

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 totalité d'une base de connaissances

Paramètres

<i>bdc</i>	: 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 toutes les propositions qui sont vraies

Paramètres

<i>listProp</i>	: liste de toutes les propositions
<i>BDVerite</i>	: base de vérité que l'on veut créer (mettre à NULL)

Renvoie

Premisse (p. 6) : renvoie la **Premisse** (p. 6) qui a 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 totalité d'une base de connaissance

Paramètres

<i>bdc</i>	: 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

<i>bdc</i>	: 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

<i>bdc</i>	: 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 conclusion d'une des règle de la bdc

Paramètres

<i>bdc</i>	: base de connaissance dans laquelle on veut chercher
<i>prop</i>	: pointeur sur la proposition que l'on veut tester

Renvoie

true : si la proposition a été trouvé dans la conclusion d'une règle

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 vérité et de la base de connaissances les propositions qui sont vraies

Paramètres

<i>baseVerite</i>	: la liste chaîné de Porposition qui sont vraie
<i>bdc</i>	: base de connaissance, liste chaîné des regles qui permettent de déduire des proposition vraie

Renvoi

Premisse (p. 6) : une liste chaînée 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 la création 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 Connaissance
- **BDConnaissances WriteBDC** (**BDConnaissances** bdc, **Premisse** *listeProposition, char chemin↔ Fichier[])
écrit 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 Connaissance.

4.3.1 Description détaillée

fichier contenant les implémentations des fonctions liée à la création 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 Connaissance

Paramètres

<i>listeProposition</i>	: liste de toute les proposition
<i>bdc</i>	: base de connaissance, liste chaîné des regles qui permettent de déduire des proposition vraie
<i>cheminFichier</i>	: chemin vers le fichier 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[] )
```

écrit 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 Connaissance.

Paramètres

<i>listeProposition</i>	: liste de toute les proposition
<i>bdc</i>	: base de connaissance, liste chaîné des regles qui permettent de déduire des proposition vraie
<i>cheminFichier</i>	: chemin vers le fichier 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 la création de la base de connaissances via la lecture de fichier

Fonctions

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

4.4.1 Description détaillée

fichier contenant les prototype des fonctions liée à la création de la base de connaissances via la lecture 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 connaissances et l'ajoute à la base de Connaissance

Paramètres

<i>listeProposition</i>	: liste de toute les proposition
<i>bdc</i>	: base de connaissance, liste chaîné des regles qui permettent de déduire des proposition vraie
<i>cheminFichier</i>	: chemin vers le fichier 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[ ] )
```

écrit 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 Connaissance.

Paramètres

<i>listeProposition</i>	: liste de toute les proposition
<i>bdc</i>	: base de connaissance, liste chaîné des regles qui permettent de déduire des proposition vraie
<i>cheminFichier</i>	: chemin vers le fichier 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 programme
- int **acquisitionEntierSecurise** ()
acquiert auprès de l'utilisateur un entier de manière sécurisé
- int **acquisitionEntierSansMessageAvecConsigne** (int min, int max, char consigne[])
acquiert un entier saisi par l'utilisateur avec contrôle d'erreur 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 écrite

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[] )
```

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

Paramètres

<i>min</i>	: valeur minimum que l'utilisateur doit saisir
<i>max</i>	: valeur maximum que l'utilisateur doit saisir
<i>consigne</i>	: consigne à afficher à l'utilisateur

Renvoie

int : revoie la valeur saisi par l'utilisateur et contrôlée

Auteur

Florian CLOAREC && Carlo AZANCOTH

Définition à la ligne 128 du fichier interface.c.

4.5.2.2 acquisitionEntierSecurise()

```
int acquisitionEntierSecurise ( )
```

acquète 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

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

Renvoie

Premisse (p. 6) : renvoie une liste chaînée 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

<i>cheminFicher</i>	: chaine de caractere qui contient le chemin vers le fichier 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

<i>proposition</i>	: 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 programme
- int **acquisitionEntierSecurise** ()
acquiert auprès de l'utilisateur un entier de manière sécurisé
- int **acquisitionEntierSansMessageAvecConsigne** (int min, int max, char consigne[])
acquiert un entier saisi par l'utilisateur avec contrôle d'erreur 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 écrite

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[] )
```

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

Paramètres

<i>min</i>	: valeur minimum que l'utilisateur doit saisir
<i>max</i>	: valeur maximum que l'utilisateur doit saisir
<i>consigne</i>	: consigne à afficher à l'utilisateur

Renvoie

int : renvoie la valeur saisie par l'utilisateur et contrôlée

Auteur

Florian CLOAREC & Carlo AZANCOTH

Définition à la ligne 128 du fichier interface.c.

4.6.2.2 acquisitionEntierSecurise()

```
int acquisitionEntierSecurise ( )
```

acquiert 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 questions à l'utilisateur

Paramètres

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

Renvoie

Premisse (p. 6) : renvoie une liste chaînée 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

<i>cheminFicher</i>	: chaine de caractère qui contient le chemin vers le fichier 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 écrite

Paramètres

<i>proposition</i>	: 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

<i>argc</i>	: nombre d'argument
<i>argv</i>	: tableau contenant les argument

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'exécution

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 fichier

```
#include "struct.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include "proposition.h"
#include "premise.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 fichier

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/premise.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 chaînée 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 Proposition appartient à une prémisse
- **Premisse** **rechercheSupprimePremisse** (**Premisse** prem, **Proposition** *prop)
supprime une proposition dans la prémisse d'une règle
- bool **premissesEmpty** (**Premisse** prem)
vérifie si une prémisse est vide ou ne l'est pas
- **Premisse** **addPropositionUnique** (**Premisse** listProp, char description[], bool valide)
ajoute et crée la proposition dans la prémisse si elle n'y est pas déjà
- **Proposition** * **rechercheListProposition** (**Premisse** listProp, char description[])
cherche si il existe une proposition avec la meme description dans une liste de proposition
- void **deletePremisseProposition** (**Premisse** prem)
supprime une prémisse et libère de la mémoire les proposition qu'elle contient
- bool **isPremisseTrue** (**Premisse** prem)
teste si toute les proposition 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 prémissse

Paramètres

<i>prem</i>	: prémissse dans lequel on veut ajouter
<i>prop</i>	: pointeur sur la proposition à ajouter

Renvoie

Premisse (p. 6) : renvoie la pémisse

Auteur

Florian CLOAREC

Définition à la ligne 24 du fichier premise.c.

4.9.2.2 addPropositionUnique()

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

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

Paramètres

<i>listProp</i>	: Premisse (p. 6) dans lequel on veux ajouter
<i>description</i>	: descption de la proposition à ajouter
<i>validite</i>	: validite de la proposition à ajouter

Renvoie

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

Auteur

Florian CLOAREC

Définition à la ligne 162 du fichier `premise.c`.**4.9.2.3 addTailPremisse()**

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

ajoute une proposition en queue à la liste chaînée de proposition qu'est la prémissse

Paramètres

<i>prem</i>	: prémissse à qui on veut ajouter une proposition
<i>prop</i>	: proposotion à ajouter

Renvoie**Premisse** (p. 6) : renvoie la pemisse**Auteur**

Florian CLOAREC

Définition à la ligne 39 du fichier `premise.c`.**4.9.2.4 affichePremisse()**

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

affiche une Prémissse de manière récursive

Paramètres

<i>prem</i>	: prémissse que l'on veut afficher
-------------	------------------------------------

Auteur

Florian CLOAREC

Définition à la ligne 84 du fichier `premise.c`.

4.9.2.5 deletePremisse()

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

supprime de la mémoire une prémisses

Paramètres

<i>prem</i>	: prémisses que l'on veut supprimer
-------------	-------------------------------------

Auteur

Florian CLOAREC

Définition à la ligne 15 du fichier premise.c.

4.9.2.6 deletePremisseProposition()

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

supprime une prémisses et libère de la mémoire les propositions qu'elle contient

Paramètres

<i>prem</i>	: prémisses à supprimer
-------------	-------------------------

Auteur

Florian CLOAREC

Définition à la ligne 222 du fichier premise.c.

4.9.2.7 isPremisseTrue()

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

teste si toutes les propositions d'une prémisses sont vraies

Paramètres

<i>prem</i>	: prémisses 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 `premise.c`.

4.9.2.8 `premisselsEmpty()`

```
bool premisselsEmpty (
    Premisse prem )
```

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

Paramètres

<i>Prem</i>	: premisses à vérifier
-------------	------------------------

Renvoie

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

Auteur

Carlo AZANCOTH

Définition à la ligne 157 du fichier `premise.c`.

4.9.2.9 `propositionDansPremisse()`

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

teste si une Proposition appartient à une prémisse

Paramètres

<i>prem</i>	: prémisse à tester
<i>prop</i>	: proposition à rechercher

Renvoie

true : si la proposition a été trouvée dans la prémisses
false : si la proposition n'a pas été trouvée dans la prémisses

Auteur

Florian CLOAREC

Définition à la ligne 107 du fichier premise.c.

4.9.2.10 rechercheListProposition()

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

cherche si il existe une proposition avec la même description dans une liste de propositions

Paramètres

<i>listProp</i>	: prémisses à tester
<i>description</i>	: description 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 premise.c.

4.9.2.11 rechercheSupprimePremisse()

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

supprime une proposition dans la prémisses d'une règle

Paramètres

<i>prem</i>	: prémisses dont on veut supprimer une proposition
<i>prop</i>	: proposition à supprimer

Renvoie

renvoie la prémisses

Auteur

Florian CLOAREC

Définition à la ligne 123 du fichier `premise.c`.

4.10 Référence du fichier `src/premise.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 permettant de stocker un élément de la liste chaînée de Proposition qui représente la Prémisses

Définitions de type

- typedef struct **PremisseElem** **PremisseElem**
- typedef **PremisseElem** * **Premisse**

Fonctions

- void **deletePremisse** (**Premisse** prem)
supprime de la mémoire une prémisses
- **Premisse** **addHeadPremisse** (**Premisse** prem, **Proposition** *prop)
ajouter une proposition en tête d'une prémisses
- **Premisse** **addTailPremisse** (**Premisse** prem, **Proposition** *prop)
ajoute une proposition en queue à la liste chaînée de proposition qu'est la prémisses
- void **affichePremisse** (**Premisse** prem)
affiche une Prémisses de manière récursive
- bool **propositionDansPremisse** (**Premisse** prem, **Proposition** *prop)
teste si une Proposition appartient à une prémisses
- **Premisse** **rechercheSupprimePremisse** (**Premisse** prem, **Proposition** *prop)
supprime une proposition dans la prémisses d'une règle
- bool **premissesEmpty** (**Premisse** prem)
vérifie si une prémisses est vide ou ne l'est pas
- **Premisse** **addPropositionUnique** (**Premisse** listProp, char description[], bool valide)
ajoute et crée la proposition dans la prémisses si elle n'y est pas déjà
- **Proposition** * **rechercheListProposition** (**Premisse** listProp, char description[])
cherche si il existe une proposition avec la même description dans une liste de proposition
- void **deletePremisseProposition** (**Premisse** prem)
supprime une prémisses et libère de la mémoire les propositions qu'elle contient
- bool **isPremisseTrue** (**Premisse** prem)
teste si toutes les propositions d'une prémisses sont vraies

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 premise.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 prémisses

Paramètres

<i>prem</i>	: prémisses dans lequel on veut ajouter
<i>prop</i>	: pointeur sur la proposition à ajouter

Renvoie

Premisse (p. 6) : renvoie la prémisses

Auteur

Florian CLOAREC

Définition à la ligne 24 du fichier `premise.c`.

4.10.3.2 `addPropositionUnique()`

```
Premisse addPropositionUnique (
    Premisse listProp,
    char description[],
    bool valide )
```

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

Paramètres

<i>listProp</i>	: Premisse (p. 6) dans lequel on veut ajouter
<i>description</i>	: description de la proposition à ajouter
<i>valide</i>	: valide de la proposition à ajouter

Renvoie

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

Auteur

Florian CLOAREC

Définition à la ligne 162 du fichier `premise.c`.

4.10.3.3 `addTailPremisse()`

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

ajoute une proposition en queue à la liste chaînée de propositions qu'est la prémisses

Paramètres

<i>prem</i>	: prémisses à qui on veut ajouter une proposition
<i>prop</i>	: proposition à ajouter

Renvoie

Premisse (p. 6) : renvoie la prémisses

Auteur

Florian CLOAREC

Définition à la ligne 39 du fichier `premise.c`.

4.10.3.4 affichePremisse()

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

affiche une Prémisses de manière récursive

Paramètres

<i>prem</i>	: prémisses que l'on veut afficher
-------------	------------------------------------

Auteur

Florian CLOAREC

Définition à la ligne 84 du fichier `premise.c`.

4.10.3.5 deletePremisse()

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

supprime de la mémoire une prémisses

Paramètres

<i>prem</i>	: prémisses que l'on veut supprimer
-------------	-------------------------------------

Auteur

Florian CLOAREC

Définition à la ligne 15 du fichier `premise.c`.

4.10.3.6 deletePremisseProposition()

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

supprime une prémisses et libère de la mémoire les propositions qu'elle contient

Paramètres

<i>prem</i>	: prémisses à supprimer
-------------	-------------------------

Auteur

Florian CLOAREC

Définition à la ligne 222 du fichier `premise.c`.

4.10.3.7 isPremisseTrue()

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

teste si toutes les propositions d'une prémisses sont vraies

Paramètres

<i>prem</i>	: prémisses à tester
-------------	----------------------

Renvoie

true : renvoie 1 si toutes les propositions de la prémisses sont vraies
false : renvoie 0 sinon

Auteur

Florian CLOAREC

Définition à la ligne 231 du fichier `premise.c`.

4.10.3.8 premisselsEmpty()

```
bool premiseIsEmpty (
    Premisse prem )
```

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

Paramètres

<i>Prem</i>	: prémisses à vérifier
-------------	------------------------

Renvoie

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

Auteur

Carlo AZANCOTH

Définition à la ligne 157 du fichier premise.c.

4.10.3.9 propositionDansPremisse()

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

teste si une Proposition appartient à une prémisses

Paramètres

<i>prem</i>	: prémisses à tester
<i>prop</i>	: proposition à rechercher

Renvoie

true : si la proposition a été trouvée dans la prémisses
false : si la proposition n'a pas été trouvée dans la prémisses

Auteur

Florian CLOAREC

Définition à la ligne 107 du fichier premise.c.

4.10.3.10 rechercheListProposition()

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

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

Paramètres

<i>listProp</i>	: premisses à tester
<i>description</i>	: description 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 premisses.c.

4.10.3.11 rechercheSupprimePremisse()

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

supprime une proposition dans la prémisses d'une règle

Paramètres

<i>prem</i>	: prémisses dont on veut supprimer une proposition
<i>prop</i>	: proposition à supprimer

Renvoie

renvoie la prémisses

Auteur

Florian CLOAREC

Définition à la ligne 123 du fichier premisses.c.

4.11 Référence du fichier src/proposition.c

fichier contenant les implémentations des fonctions associées à la structure de données Proposition

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

Fonctions

- **Proposition * newProposition** (char valueProposition[], bool valide)
- structure de donnée **Proposition** (p. 7) qui est une chaîne de caractère*
- void **deleteProposition** (**Proposition** *propositionToDelete)
- supprime de la mémoire une variable de type proposition*
- void **afficheProposition** (**Proposition** *proposition)
- affiche le contenu d'une proposition*
- void **setValide** (**Proposition** *prop, bool val)
- définit la valeur de la sous variable valide d'une proposition*

4.11.1 Description détaillée

fichier contenant les implémentations des fonctions associées à la structure de donnée Proposition

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 afficheProposition()

```
void afficheProposition (
    Proposition * proposition )
```

affiche le contenu d'une proposition

Paramètres

<i>proposition</i>	: 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

<i>propositionToDelete</i>	: 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 valide )
```

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

Auteur

Florian CLOAREC

crée une variable de type proposiont

Paramètres

<i>valuePropositon[]</i>	: chaine de caractère qui contient la valuer de la propositon à créer
<i>valide</i>	: 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

<i>prop</i>	: proposition que l'on veut modifier
<i>val</i>	: 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 associées à la structure de donnée proposition ainsi que la déclaration de la structure de donnée proposition elle-même

Structures de données

- struct **Proposition**
structure de donnée permettant de stocker ou une Proposition

Définitions de type

- typedef struct **Proposition** **Proposition**

Fonctions

- **Proposition** * **newProposition** (char valueProposition[], bool valide)
*structure de donnée **Proposition** (p. 7) qui est une chaîne de caractère*
- void **deleteProposition** (**Proposition** *propositionToDelete)
supprime de la mémoire une variable de type proposition
- void **afficheProposition** (**Proposition** *proposition)
affiche le contenu d'une proposition
- 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 déclaration 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 affichePropositon()

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

afficher le contenu d'une proposition

Paramètres

<i>proposition</i>	: 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

<i>propositionToDelete</i>	: 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 valide )
```

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

Auteur

Florian CLOAREC

crée une variable de type proposition

Paramètres

<i>valueProposition[]</i>	: chaîne de caractère qui contient la valeur de la proposition à créer
<i>valide</i>	: 1 si la proposition est vraie, 0 si elle est fausse

Renvoie

Proposition (p. 7) : pointeur sur la proposition qui vient d'être créée

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

<i>prop</i>	: proposition que l'on veut modifier
<i>val</i>	: 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 conclusion d'une règle
- void **afficheRegle (Regle *regle)**
affiche une règle
- bool **ReglePremisselsEmpty (Regle *regleAVerif)**
vérifie si la premisses d'une règle est vide ou ne l'est pas
- **Regle * supprimePropositionPremisseRegle (Regle *regle, Proposition *prop)**
supprime la proposition de la prémisses d'une règle
- **Proposition * conclutionRegle (Regle *regle)**
revoie la conclusion d'une règle
- **Premisse instertHeadPremisseRegle (Regle *regle, Premisse listProp, char *description)**
ajoute en tete une proposition à la prémisses 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 conclusion d'une règle

Paramètres

<i>regle</i>	: règle pour lequel on définit la conclusion
<i>listProp</i>	: liste de toute les proposition
<i>description</i>	: 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

<i>regle</i>	: pointeur vers la règle à afficher
--------------	-------------------------------------

Auteur

Carlo AZANCOTH

Définition à la ligne 45 du fichier regle.c.

4.13.2.3 conclusionRegle()

```
Proposition* conclusionRegle (
    Regle * regle )
```

revoie la conclusion d'une règle

Paramètres

<i>regle</i>	: 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

<i>pListProp</i>	: pointeur sur la liste de toute les proposition qui on déjà été utilisé
<i>descriptionPremisse</i>	: tableau de chaine de caractère contenant toutes les descriptions des prémisses à ajouter à la règle
<i>nbElemPremisse</i>	: nombre d'élément que l'on veut mettre dans la prémisses
<i>descriptionConclusion</i>	: description à donner à la conclusion

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

<i>regleToDelete</i>	: regle que l'on veut supprimer
----------------------	---------------------------------

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 prémisses d'une règle

Paramètres

<i>regle</i>	: règle à laquelle on veux ajouter une proposition
<i>listProp</i>	: liste de toute les propositions
<i>description</i>	: 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 ReglePremisselsEmpty()

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

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

Paramètres

<i>regleAVerif</i>	: regle ou se trouve la premisses à vérifier
--------------------	--

Renvoie

1 : si la premisses est vide

0 : si la premisses 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

<i>regle</i>	: règle dans laquelle on veut supprimer
<i>prop</i>	: proposition à rechercher et supprimer

Renvoie

pointeur sur la règle avec laquelle on a travaillé

Auteur

Florian CLOAREC

Définition à la ligne 65 du fichier regle.c.

4.14 Référence du fichier src/regle.h

fichier contenant les prototypes liés à la structure de données **Regle** (p. 8) ainsi que les déclarations des structures de données **Regle** (p. 8) et ses composées

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

Structures de données

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

Définitions de type

— typedef struct **Regle** **Regle**

Fonctions

- **Regle** * **newRegle** ()
crée une règle vide
- void **deleteRegle** (**Regle** *regleToDelete)
supprime 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 **ReglePremisselsEmpty** (**Regle** *regleAVerif)
vérifie si la premissse d'une règle est vide ou ne l'est pas
- **Regle** * **supprimePropositionPremisseRegle** (**Regle** *regle, **Proposition** *prop)
supprime la proposition de la prémissse d'une règle
- **Proposition** * **conclusionRegle** (**Regle** *regle)
revoie la conclution d'une règle
- **Premisse** **instertHeadPremisseRegle** (**Regle** *regle, **Premisse** listProp, char *description)
ajoute en tete une propostion à la prémissse 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 prototype 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 conclusion d'une règle

Paramètres

<i>regle</i>	: regle pour lequel on définit la conclusion
<i>listProp</i>	: liste de toute les proposition
<i>description</i>	: 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

<i>regle</i>	: pointeur vers la règle à afficher
--------------	-------------------------------------

Auteur

Carlo AZANCOTH

Définition à la ligne 45 du fichier regle.c.

4.14.3.3 conclusionRegle()

```
Proposition* conclusionRegle (
    Regle * regle )
```

revoie la conclusion d'une règle

Paramètres

<i>regle</i>	: 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 chaîne de caractère

Paramètres

<i>pListProp</i>	: pointeur sur la liste de toutes les propositions qui ont déjà été utilisées
<i>descriptionPremisse</i>	: tableau de chaînes de caractères contenant toutes les descriptions des prémisses à ajouter à la règle
<i>nbElemPremisse</i>	: nombre d'éléments que l'on veut mettre dans la prémisse
<i>descriptionConclusion</i>	: description à donner à la conclusion

Renvoie

Regle* : renvoie un pointeur sur la règle qui a été créée

Auteur

Florian CLOAREC

Définition à la ligne 91 du fichier regle.c.

4.14.3.5 deleteRegle()

```

void deleteRegle (
    Regle * regleToDelete )

```

supprime une règle ainsi que tous ses composants

Paramètres

<i>regleToDelete</i>	: règle que l'on veut supprimer
----------------------	---------------------------------

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

<i>regle</i>	: règle à laquelle on veut ajouter une proposition
<i>listProp</i>	: liste de toute les propositions
<i>description</i>	: 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 ReglePremisselsEmpty()

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

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

Paramètres

<i>regleAVerif</i>	: regle ou se trouve la premissse à vérifier
--------------------	--

Renvoie

- 1 : si la premissse est vide
- 0 : si la premissse 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 prémissse d'une règle

Paramètres

<i>regle</i>	: règle dans laquelle on veut supprimer
<i>prop</i>	: proposition à rechercher et supprimer

Renvoie

pointeur sur la règle avec laquelle on a 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

GNU GENERAL PUBLIC LICENSE

Index

- acquisitionEntierSansMessageAvecConsigne
 - interface.c, 26
 - interface.h, 29
- acquisitionEntierSecurise
 - interface.c, 27
 - interface.h, 30
- addConclusion
 - regle.c, 53
 - regle.h, 58
- addHeadBDC
 - BDConnaissances.c, 12
 - BDConnaissances.h, 18
- addHeadPremisse
 - premise.c, 35
 - premise.h, 41
- addPropositionUnique
 - premise.c, 35
 - premise.h, 42
- addRegleBDC
 - BDConnaissances.c, 12
 - BDConnaissances.h, 18
- addTailPremisse
 - premise.c, 36
 - premise.h, 42
- afficheBDC
 - BDConnaissances.c, 13
 - BDConnaissances.h, 19
- affichePremisse
 - premise.c, 36
 - premise.h, 43
- affichePropositon
 - proposition.c, 47
 - proposition.h, 50
- afficheRegle
 - regle.c, 53
 - regle.h, 60
- BDConnaissances
 - BDConnaissances.h, 17
- BDConnaissances.c
 - addHeadBDC, 12
 - addRegleBDC, 12
 - afficheBDC, 13
 - createBDVerite, 13
 - deleteAllBDC, 14
 - deleteHeadBDC, 14
 - isEmptyBDC, 14
 - isPropositionInConclusion, 15
 - moteurDIInference, 15
- BDConnaissances.h
 - addHeadBDC, 18
 - addRegleBDC, 18
 - afficheBDC, 19
 - BDConnaissances, 17
 - BDConnaissancesElem, 17
 - createBDVerite, 19
 - deleteAllBDC, 19
 - deleteHeadBDC, 20
 - isEmptyBDC, 20
 - isPropositionInConclusion, 21
 - moteurDIInference, 21
- BDConnaissancesElem, 5
 - BDConnaissances.h, 17
 - suivant, 5
 - valeur, 5
- BDConnaissanceselem, 6
- conclusion
 - Regle, 9
- conclusionRegle
 - regle.c, 54
 - regle.h, 60
- createBDVerite
 - BDConnaissances.c, 13
 - BDConnaissances.h, 19
- createRegle
 - regle.c, 54
 - regle.h, 61
- deleteAllBDC
 - BDConnaissances.c, 14
 - BDConnaissances.h, 19
- deleteHeadBDC
 - BDConnaissances.c, 14
 - BDConnaissances.h, 20
- deletePremisse
 - premise.c, 36
 - premise.h, 43
- deletePremisseProposition
 - premise.c, 37
 - premise.h, 44
- deleteProposition
 - proposition.c, 47
 - proposition.h, 51
- deleteRegle
 - regle.c, 55
 - regle.h, 61
- description
 - Proposition, 8

- elemSuivant
 - PremisseElem, 7
- fichier.c
 - ReadBDC, 23
 - WriteBDC, 23
- fichier.h
 - ReadBDC, 24
 - WriteBDC, 25
- genereBDVerite
 - interface.c, 27
 - interface.h, 30
- instertHeadPremisseRegle
 - regle.c, 55
 - regle.h, 62
- interface.c
 - acquisitionEntierSansMessageAvecConsigne, 26
 - acquisitionEntierSecurise, 27
 - genereBDVerite, 27
 - menuPrincipal, 28
 - systemExpert, 28
 - verificationPropositionAvecMessage, 28
- interface.h
 - acquisitionEntierSansMessageAvecConsigne, 29
 - acquisitionEntierSecurise, 30
 - genereBDVerite, 30
 - menuPrincipal, 31
 - systemExpert, 31
 - verificationPropositionAvecMessage, 31
- isEmptyBDC
 - BDConnaissances.c, 14
 - BDConnaissances.h, 20
- isPremisseTrue
 - premise.c, 37
 - premise.h, 44
- isPropositionInConclusion
 - BDConnaissances.c, 15
 - BDConnaissances.h, 21
- main
 - main.c, 32
- main.c
 - main, 32
- main.h
 - TAILLE_MAXI_PROPOSITION, 34
- menuPrincipal
 - interface.c, 28
 - interface.h, 31
- moteurDIference
 - BDConnaissances.c, 15
 - BDConnaissances.h, 21
- newProposition
 - proposition.c, 48
 - proposition.h, 51
- newRegle
 - regle.c, 56
- regle.h, 62
- Premisse, 6
 - premise.h, 41
- premise
 - Regle, 9
- premise.c
 - addHeadPremisse, 35
 - addPropositionUnique, 35
 - addTailPremisse, 36
 - affichePremisse, 36
 - deletePremisse, 36
 - deletePremisseProposition, 37
 - isPremisseTrue, 37
 - premisselsEmpty, 38
 - propositionDansPremisse, 38
 - rechercheListProposition, 39
 - rechercheSupprimePremisse, 39
- premise.h
 - addHeadPremisse, 41
 - addPropositionUnique, 42
 - addTailPremisse, 42
 - affichePremisse, 43
 - deletePremisse, 43
 - deletePremisseProposition, 44
 - isPremisseTrue, 44
 - Premisse, 41
 - PremisseElem, 41
 - premisselsEmpty, 44
 - propositionDansPremisse, 45
 - rechercheListProposition, 45
 - rechercheSupprimePremisse, 46
- PremisseElem, 6
 - elemSuivant, 7
 - premise.h, 41
 - valeur, 7
- premisselsEmpty
 - premise.c, 38
 - premise.h, 44
- Proposition, 7
 - description, 8
 - proposition.h, 50
 - validite, 8
- proposition.c
 - affichePropositon, 47
 - deleteProposition, 47
 - newProposition, 48
 - setValidite, 48
- proposition.h
 - affichePropositon, 50
 - deleteProposition, 51
 - newProposition, 51
 - Proposition, 50
 - setValidite, 51
- propositionDansPremisse
 - premise.c, 38
 - premise.h, 45
- ReadBDC

- fichier.c, 23
- fichier.h, 24
- rechercheListProposition
 - premise.c, 39
 - premise.h, 45
- rechercheSupprimePremisse
 - premise.c, 39
 - premise.h, 46
- Regle, 8
 - conclusion, 9
 - premise, 9
 - regle.h, 58
- regle.c
 - addConclusion, 53
 - afficheRegle, 53
 - conclusionRegle, 54
 - createRegle, 54
 - deleteRegle, 55
 - insertHeadPremisseRegle, 55
 - newRegle, 56
 - ReglePremisselsEmpty, 56
 - supprimePropositionPremisseRegle, 56
- regle.h
 - addConclusion, 58
 - afficheRegle, 60
 - conclusionRegle, 60
 - createRegle, 61
 - deleteRegle, 61
 - insertHeadPremisseRegle, 62
 - newRegle, 62
 - Regle, 58
 - ReglePremisselsEmpty, 62
 - supprimePropositionPremisseRegle, 63
- ReglePremisselsEmpty
 - regle.c, 56
 - regle.h, 62
- setValidite
 - proposition.c, 48
 - proposition.h, 51
- src/BDConnaissances.c, 11
- src/BDConnaissances.h, 16
- src/fichier.c, 22
- src/fichier.h, 24
- src/interface.c, 25
- src/interface.h, 29
- src/main.c, 32
- src/main.h, 33
- src/premise.c, 34
- src/premise.h, 40
- src/proposition.c, 46
- src/proposition.h, 49
- src/regle.c, 52
- src/regle.h, 57
- src/struct.h, 63
- suivant
 - BDConnaissancesElem, 5
- supprimePropositionPremisseRegle
 - regle.c, 56
 - regle.h, 63
- systemExpert
 - interface.c, 28
 - interface.h, 31
- TAILLE_MAXI_PROPOSITION
 - main.h, 34
- valeur
 - BDConnaissancesElem, 5
 - PremisseElem, 7
- validite
 - Proposition, 8
- verificationPropositionAvecMessage
 - interface.c, 28
 - interface.h, 31
- WriteBDC
 - fichier.c, 23
 - fichier.h, 25