# IFB_projet

V1.0

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 Carte

represente une carte d'un jeu de 32 cartes

`#include <main.h>`

Graphe de collaboration de Carte:

### Champs de données

- Couleur couleur
- Valeur valeur

### 3.1.1 Description détaillée

represente une carte d'un jeu de 32 cartes

Définition à la ligne 133 du fichier main.h.

### 3.1.2 Documentation des champs

#### 3.1.2.1 couleur

`Couleur couleur`

Définition à la ligne 135 du fichier main.h.

#### 3.1.2.2 valeur

```
Valeur valeur
```

type Couleur, donne la famille Ã  laquel la carte apprartient

Définition à la ligne 136 du fichier main.h.

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

## 3.2 Référence de la structure Contrat

represente un contrat Ã  la belote coichÃ©e

```
#include <main.h>
```

Graphe de collaboration de Contrat:

### Champs de données

— Joueur preneur
— NbPoint nbPoint
— Couleur atout
— Coinche coinche

### 3.2.1 Description détaillée

represente un contrat Ã  la belote coichÃ©e

Permet d'avoir toutes les information concernant un contrat stockÃ© dans une seuil et mÃªme variable

Définition à la ligne 148 du fichier main.h.

### 3.2.2 Documentation des champs

#### 3.2.2.1 atout

```
Couleur atout
```

type NbPoint, nombre de point du contrat

Définition à la ligne 152 du fichier main.h.

#### 3.2.2.2 coinche

```
Coinche coinche
```

type Couleur, couleur de l'atout qui Ã Ã©tÃ© prise pour le contrat

Définition à la ligne 153 du fichier main.h.

#### 3.2.2.3 nbPoint

```
NbPoint nbPoint
```

type Joueur, joueur qui a pris le contrat

Définition à la ligne 151 du fichier main.h.

#### 3.2.2.4 preneur

```
Joueur preneur
```

Définition à la ligne 150 du fichier main.h.

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

# Chapitre 4

# Documentation des fichiers

## 4.1 Référence du fichier acquisition.c

ficher source contenant les fonctions relative a l'acquisirion

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

Graphe des dépendances par inclusion de acquisition.c:

## 4.2 Référence du fichier acquisition.h

ficher header contenant les prototype des fonctions relative a l'acquisirion

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

### Fonctions

- int acquisitionEntierAvecMessage (int min, int max)
  *renvois au progammme un entier saisi par l'utilisateur avec message d'erreur entre 2 entiers*
- int acquisitionEntierSansMessage (int min, int max)
  *accisition d'un entier saisi par l'utilisateur sans message d'erreur entre 2 entiers*
- char acquisitionOuiNonSansMessage ()
  *accisition O/N 'oui ou non' saisi par l'utilisateur sans message d'erreur*
- int acquisitionEntierSansMessageAvecConsigne (int min, int max, char consigne[ ])
  *accisition d'un entier saisi par l'utilisateur sans message d'erreur entre 2 entiers avec une consigne paramétrable*
- void acquisitionPseudoAvecMessage (char ∗pointeurPseudo, char instruction[ ], int type)
  *acquisition d'un pseudo saisi par l'utilisateur avec message d'erreur*
- int acquisitionEntierSecurise ()
  *fonction qui fait l'acquisition securisée d'un nombre*

### 4.2.1 Description détaillée

ficher header contenant les prototype des fonctions relative a l'acquisirion

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.2.2 Documentation des fonctions

#### 4.2.2.1 acquisitionEntierAvecMessage()

```
int acquisitionEntierAvecMessage (
            int min,
            int max )
```

renvois au progammme un entier saisi par l'utilisateur avec message d'erreur entre 2 entiers

**Paramètres**

| | |
|---|---|
| *min* | : borne inférieur de l'ensemble dans lequel la valeur saisi doit ce trouver |
| *max* | : borne superieur de l'ensemble dans lequel la valeur saisi doit ce trouver |

**Renvoie**

la valeur saise par l'utilisateur et vérifier

Définition à la ligne 11 du fichier acquisition.c.

Voici le graphe d'appel pour cette fonction :

#### 4.2.2.2 acquisitionEntierSansMessage()

```
int acquisitionEntierSansMessage (
            int min,
            int max )
```

accisition d'un entier saisi par l'utilisateur sans message d'erreur entre 2 entiers

**Paramètres**

| | |
|---|---|
| *min* | : borne inférieur de l'ensemble dans lequel la valeur saisi doit ce trouver |
| *max* | : borne superieur de l'ensemble dans lequel la valeur saisi doit ce trouver |

**Renvoie**

la valeure entiere saise par l'utilisateur et vérifier

Définition à la ligne 24 du fichier acquisition.c.

Voici le graphe d'appel pour cette fonction :

#### 4.2.2.3 acquisitionEntierSansMessageAvecConsigne()

```
int acquisitionEntierSansMessageAvecConsigne (
            int min,
```

```
            int max,
            char consigne[] )
```

accisition d'un entier saisi par l'utilisateur sans message d'erreur entre 2 entiers avec une consigne paramétrable

**Paramètres**

| | |
|---|---|
| *min* | : borne inférieur de l'ensemble dans lequel la valeur saisi doit ce trouver |
| *max* | : borne superieur de l'ensemble dans lequel la valeur saisi doit ce trouver |
| *consigne* | : message affiché avant a l'utilisateur pour lui indiquer ce qu'il doit saisir |

**Renvoie**

la valeure entiere saise par l'utilisateur et vérifier

Définition à la ligne 42 du fichier acquisition.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.2.2.4 acquisitionEntierSecurise()

```
int acquisitionEntierSecurise ( )
```

fonction qui fait l'acquisition securisée d'un nombre

**Renvoie**

renvoie le nombre saisi par l'utilisateur et -1 si aucun nombre a été saisi

Définition à la ligne 94 du fichier acquisition.c.

Voici le graphe des appelants de cette fonction :

#### 4.2.2.5 acquisitionOuiNonSansMessage()

```
char acquisitionOuiNonSansMessage ( )
```

accisition O/N 'oui ou non' saisi par l'utilisateur sans message d'erreur

**Renvoie**

le caratere saisi par l'utilisateur, 'O' ou 'N' qui a ete verifie

Définition à la ligne 33 du fichier acquisition.c.

#### 4.2.2.6 acquisitionPseudoAvecMessage()

```
void acquisitionPseudoAvecMessage (
            char * pointeurPseudo,
            char instruction[],
            int type )
```

acquisition d'un pseudo saisi par l'utilisateur avec message d'erreur

**Paramètres**

| | |
|---|---|
| *PPseudo* | pointeur renvoyant vers le premier carractère du pseudo à modifier |
| *instruction* | phrase à affichier pour donner l'instruction à l'utilisateur |
| *type* | : vaut 1 si la fonction est appelée dans la fonction paramètre |

**Renvoie**

void

Définition à la ligne 52 du fichier acquisition.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

## 4.3 Référence du fichier affichage.c

ficher source contenant les fonctions relative à l'affichage

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

Graphe des dépendances par inclusion de affichage.c:

### Fonctions

- int afficheMenuPrincipal (int type)
  *affiche le logo du jeu puis le menu principal*
- int afficheInterfacePli (Carte dernierPli[ ], Carte pli[ ], char ∗pseudo[ ], Carte cartesEnMain[ ], Contrat contratActuel, char message[ ], Joueur utilisteur, Joueur dernierVainqueur, int score[ ], int pointManche[ ], int type)
  *affiche l'interface de jeu durant un pli*
- void modifieTailleFenetre (int nbLigneFenetre, int nbColloneFentre)
  *modifie la taille de la fenetre dans laquel le programme s'execute*
- void afficheSousMenus (char phrase[ ], char intitule[ ])
  *affiche les différents sous menus*
- void afficheMain (Carte carte[ ])
  *affiche les carte qu'un joueur a en main*
- void afficheContrat (Contrat contrat, char ∗pseudo[ ], int version)
  *affiche un contrat*
- void afficheMenuSelection (char intitule[ ], char phrase[ ], int sautDeLigne)
  *affiche dans un cadre plusieurs chaines de carractères avec un certain nombre de lignes sautée entre chacunes d'elles*
- Contrat proposeContratUtilisateur (Contrat dernierContrat, Joueur parle, Carte ∗pCarteMain)
  *gère le choix du contrat par l'utilisateur*

#### 4.3.1 Description détaillée

ficher source contenant les fonctions relative à l'affichage

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.3.2 Documentation des fonctions

#### 4.3.2.1 afficheContrat()

```
void afficheContrat (
            Contrat contrat,
            char * pseudo[],
            int version )
```

affiche un contrat

**Paramètres**

| | |
|---|---|
| *Contrat* | contrat : contrat a afficher |
| *char* | ∗pseudo[]: tableau de pointeur contenant les pseudo des différents joueurs |
| *int* | version : 1 pour la version belle et de grande taille et 2 pour la version courte |

**Renvoie**

void

Définition à la ligne 247 du fichier affichage.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.3.2.2 afficheInterfacePli()

```
afficheInterfacePli (
            Carte dernierPli[],
            Carte pli[],
            char * pseudo[],
            Carte cartesEnMain[],
            Contrat contratActuel,
            char message[],
            Joueur utilisteur,
            Joueur dernierVainqueur,
            int score[],
            int pointManche[],
            int type )
```

affiche l'interface de jeu durant un pli

**Paramètres**

| | |
|---|---|
| *Carte* | dernierPli[] : tableau contenant les 4 cartes du dernier plis |
| *Carte* | pli[] : tableau contenant les cartes du pli en cours, mettre 0, 0 si il n'y a pas de carte |
| *char* | ∗pseudo[] : tableau contenant les pseudo des 4 joueur |
| *Carte* | cartesEnMain[] : tableau contenant les carte dans la main de l'utilisateur |
| *Contrat* | contratActuel : contrat qui est en cour dans cette manche |

**Paramètres**

| | |
|---|---|
| *char* | message[500] |
| *Joueur* | utilisateur : donne la position de l'utilisateur, mettre SANS_Joueur pour faire une partie avec uniquement des ordinateur |
| *Joueur* | dernierVainqueur : joueur aylant gagné le dernier pli |
| *int* | score[] : tableau qui contient les score de la partie |
| *int* | pointManche[] : tableau qui contient les points de la manche |
| *int* | type : version de la fonction : 0 pour un affichage et une acquisition et 1 pour l'acquisition seulle |

**Renvoie**

int : valeur choisie par l'utilisateur

< Formatage des chaine de caracère relative aux pseudo

< Formatage des chaine de caractère relative au message

< Formatage des chaine de caractère relative aux cartes

< Formatage des chaine de caractère relative au contrat

Définition à la ligne 59 du fichier affichage.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.3.2.3 afficheMain()

```
void afficheMain (
            Carte carte[] )
```

affiche les carte qu'un joueur a en main

**Paramètres**

| | |
|---|---|
| *Carte* | carte[] : tableau contanant les carte |

**Renvoie**

void

Définition à la ligne 227 du fichier affichage.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.3.2.4 afficheMenuPrincipal()

```
int afficheMenuPrincipal (
            int type )
```

affiche le logo du jeu puis le menu principal

**Paramètres**

| | |
|---|---|
| *type* | quel type de menu est souhaité |

**Renvoie**

la valeur de l'action que l'utilisateur decide de faire

Définition à la ligne 11 du fichier affichage.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.3.2.5 afficheMenuSelection()

```
void afficheMenuSelection (
             char intitule[],
             char phrase[],
             int sautDeLigne )
```

affiche dans un cadre plusieurs chaines de carractères avec un certain nombre de lignes sautée entre chacunes d'elles

**Paramètres**

| | |
|---|---|
| *char* | intitule []: titre du du cadre |
| *char* | phrase[] : les chaines de carractère séparés par des points virgules |
| *int* | sautDeLigne : nombre de lignes à sauter entre chaque chaine de carractères |

**Renvoie**

void

< contrôle le centrage du titre du sous-menu

< effacement de la chaine de carrateres

<efface l'écran et affiche le sous-menu avec le texte correspondant

Définition à la ligne 269 du fichier affichage.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.3.2.6 afficheSousMenus()

```
void afficheSousMenus (
             char phrase[],
             char intitule[] )
```

affiche les différents sous menus

**Paramètres**

| | |
|---|---|
| *phrase* | phrase correspondant au sous menu choisi |

**Renvoie**

void

< contrôle le centrage du titre du sous-menu

< découpe la phrase pour pouvoir l'afficher

<efface l'écran et affiche le sous-menu avec le texte correspondant

Définition à la ligne 186 du fichier affichage.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.3.2.7 modifieTailleFenetre()

```
void modifieTailleFenetre (
            int nbLigneFenetre,
            int nbColloneFentre )
```

modifie la taille de la fenetre dans laquel le programme s'execute

**Paramètres**

| | |
|---|---|
| *nbLingeFenetre* | : nombre de ligne de la fenetre |
| *nbColloneFenetre* | : nombre de collone de la fenetre |

**Renvoie**

void

Définition à la ligne 177 du fichier affichage.c.

Voici le graphe des appelants de cette fonction :

#### 4.3.2.8 proposeContratUtilisateur()

```
Contrat proposeContratUtilisateur (
            Contrat dernierContrat,
            Joueur parle,
            Carte * pCarteMain )
```

gère le choix du contrat par l'utilisateur

**Paramètres**

| | |
|---|---|
| *Contrat* | dernierContrat : dernier contrat proposé |
| *Joueur* | parle : joueur qui parle |
| *Carte* | ∗pCarteMain : pointeur sur le tableau qui stocke les carte dans la main du joueur |

**Renvoie**

Contrat : nouveau contrat proposé par le joueur

**Paramètres**

| | |
|---|---|
| | |

Définition à la ligne 348 du fichier affichage.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

## 4.4 Référence du fichier affichage.h

ficher header contenant les prototypes des fonctions relative à l'affichage

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

### Fonctions

- int afficheMenuPrincipal (int type)
  *affiche le logo du jeu puis le menu principal*
- int afficheInterfacePli (Carte dernierPli[ ], Carte pli[ ], char ∗pseudo[ ], Carte cartesEnMain[ ], Contrat contratActuel, char message[ ], Joueur utilisteur, Joueur dernierVainqueur, int score[ ], int pointManche[ ], int type)
  *affiche l'interface de jeu durant un pli*
- void modifieTailleFenetre (int nbLigneFenetre, int nbColloneFentre)
  *modifie la taille de la fenetre dans laquel le programme s'execute*
- void afficheSousMenus (char phrase[ ], char intitule[ ])
  *affiche les différents sous menus*
- void afficheMain (Carte carte[ ])
  *affiche les carte qu'un joueur a en main*
- void afficheContrat (Contrat contrat, char ∗pseudo[ ], int version)
  *affiche un contrat*
- void afficheMenuSelection (char intitule[ ], char phrase[ ], int sautDeLigne)
  *affiche dans un cadre plusieurs chaines de carractères avec un certain nombre de lignes sautée entre chacunes d'elles*
- Contrat proposeContratUtilisateur (Contrat dernierContrat, Joueur parle, Carte ∗pCarteMain)
  *gère le choix du contrat par l'utilisateur*

### 4.4.1 Description détaillée

ficher header contenant les prototypes des fonctions relative à l'affichage

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.4.2 Documentation des fonctions

#### 4.4.2.1 afficheContrat()

```
void afficheContrat (
            Contrat contrat,
            char * pseudo[],
            int version )
```

affiche un contrat

**Paramètres**

| | |
|---|---|
| *Contrat* | contrat : contrat a afficher |
| *char* | ∗pseudo[]: tableau de pointeur contenant les pseudo des différents joueurs |
| *int* | version : 1 pour la version belle et de grande taille et 2 pour la version courte |

**Renvoie**

void

Définition à la ligne 247 du fichier affichage.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.4.2.2 afficheInterfacePli()

```
int afficheInterfacePli (
            Carte dernierPli[],
            Carte pli[],
            char * pseudo[],
            Carte cartesEnMain[],
            Contrat contratActuel,
            char message[],
            Joueur utilisteur,
            Joueur dernierVainqueur,
            int score[],
            int pointManche[],
            int type )
```

affiche l'interface de jeu durant un pli

**Paramètres**

| | |
|---|---|
| *Carte* | dernierPli[] : tableau contenant les 4 cartes du dernier plis |
| *Carte* | pli[] : tableau contenant les cartes du pli en cours, mettre 0, 0 si il n'y a pas de carte |
| *char* | ∗pseudo[] : tableau contenant les pseudo des 4 joueur |
| *Carte* | cartesEnMain[] : tableau contenant les carte dans la main de l'utilisateur |
| *Contrat* | contratActuel : contrat qui est en cour dans cette manche |
| *char* | message[500] |
| *Joueur* | utilisateur : donne la position de l'utilisateur, mettre SANS_Joueur pour faire une partie avec uniquement des ordinateur |
| *Joueur* | dernierVainqueur : joueur aylant gagné le dernier pli |
| *int* | score[] : tableau qui contient les score de la partie |
| *int* | pointManche[] : tableau qui contient les points de la manche |
| *int* | type : version de la fonction : 0 pour un affichage et une acquisition et 1 pour l'acquisition seulle |

**Renvoie**

int : valeur choisie par l'utilisateur

< Formatage des chaine de caracère relative aux pseudo

< Formatage des chaine de caractère relative au message

< Formatage des chaine de caractère relative aux cartes

< Formatage des chaine de caractère relative au contrat

Définition à la ligne 59 du fichier affichage.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.4.2.3 afficheMain()

```
void afficheMain (
            Carte carte[] )
```

affiche les carte qu'un joueur a en main

**Paramètres**

| | |
|---|---|
| *Carte* | carte[] : tableau contanant les carte |

**Renvoie**

void

Définition à la ligne 227 du fichier affichage.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.4.2.4 afficheMenuPrincipal()

```
int afficheMenuPrincipal (
            int type )
```

affiche le logo du jeu puis le menu principal

**Paramètres**

| | |
|---|---|
| *type* | quel type de menu est souhaité |

**Renvoie**

la valeur de l'action que l'utilisateur decide de faire

Définition à la ligne 11 du fichier affichage.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.4.2.5 afficheMenuSelection()

```
void afficheMenuSelection (
            char intitule[],
            char phrase[],
            int sautDeLigne )
```

affiche dans un cadre plusieurs chaines de carractères avec un certain nombre de lignes sautée entre chacunes d'elles

**Paramètres**

| | |
|---|---|
| *char* | intitule []: titre du du cadre |
| *char* | phrase[] : les chaines de carractère séparés par des points virgules |
| *int* | sautDeLigne : nombre de lignes à sauter entre chaque chaine de carractères |

**Renvoie**

void

< contrôle le centrage du titre du sous-menu

< effacement de la chaine de carrateres

<efface l'écran et affiche le sous-menu avec le texte correspondant

Définition à la ligne 269 du fichier affichage.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.4.2.6 afficheSousMenus()

```
void afficheSousMenus (
            char phrase[],
            char intitule[] )
```

affiche les différents sous menus

**Paramètres**

| | |
|---|---|
| *phrase* | phrase correspondant au sous menu choisi |

**Renvoie**

void

< contrôle le centrage du titre du sous-menu

< découpe la phrase pour pouvoir l'afficher

<efface l'écran et affiche le sous-menu avec le texte correspondant

Définition à la ligne 186 du fichier affichage.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.4.2.7 modifieTailleFenetre()

```
void modifieTailleFenetre (
            int nbLigneFenetre,
            int nbColloneFentre )
```

modifie la taille de la fenetre dans laquel le programme s'execute

**Paramètres**

| | |
|---|---|
| *nbLingeFenetre* | : nombre de ligne de la fenetre |
| *nbColloneFenetre* | : nombre de collone de la fenetre |

**Renvoie**

void

Définition à la ligne 177 du fichier affichage.c.

Voici le graphe des appelants de cette fonction :

#### 4.4.2.8 proposeContratUtilisateur()

```
Contrat proposeContratUtilisateur (
            Contrat dernierContrat,
            Joueur parle,
            Carte * pCarteMain )
```

gère le choix du contrat par l'utilisateur

**Paramètres**

| | |
|---|---|
| *Contrat* | dernierContrat : dernier contrat proposé |
| *Joueur* | parle : joueur qui parle |
| *Carte* | ∗pCarteMain : pointeur sur le tableau qui stocke les carte dans la main du joueur |

**Renvoie**

Contrat : nouveau contrat proposé par le joueur

**Paramètres**

| | |
|---|---|
| | |

Définition à la ligne 348 du fichier affichage.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

## 4.5 Référence du fichier autre.c

ficher source contenant les fonctions qui n'allait dans aucun autre fichier

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

Graphe des dépendances par inclusion de autre.c:

### Fonctions

- Joueur joueurSuivant (Joueur joueur)
  *passe au joueur suivant*
- int nbAleatoire (int mini, int maxi)
  *renvoit in nombre aléatoire entre deux bornes*
- void setContrat (Contrat ∗contrat, Joueur preneur, int nbPoint, Couleur atout, Coinche coinche)
  *définit un contrat*
- int pointPli (Carte pli[ ], Couleur atout, int nbCarte)
  *compte le nombre de points d'un pli*
- float ajusteEchelle (float valeur, float entreMin, float entreMax, float sortieMin, float sortieMax)
  *met a l'echelle la valeur d'une variable comprise entre deux bornes pour que la variable de retour soit compris entre deux autres bornes*
- void joue1000Partie (int nbPartie)
  *joue n partie aves uniquement des ia pour controler si tout ce passe bien et reuceillir des statistique*

### 4.5.1 Description détaillée

ficher source contenant les fonctions qui n'allait dans aucun autre fichier

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

## 4.5.2 Documentation des fonctions

### 4.5.2.1 ajusteEchelle()

```
float ajusteEchelle (
            float valeur,
            float entreMin,
            float entreMax,
            float sortieMin,
            float sortieMax )
```

met a l'echelle la valeur d'une variable comprise entre deux bornes pour que la variable de retour soit compris entre deux autres bornes

**Paramètres**

| | |
|---|---|
| *float* | valeur : valeur a mettre à l'echelle |
| *float* | entreMin : valeur minimum que peut prendre l'entrée |
| *float* | entreMax : valeur maximum que peut prendre l'entrée |
| *float* | sortieMin : valeur minimum que peut prendre la sortie |
| *float* | sortieMax : valeur maximum que peut prendre la sortie |

**Renvoie**

float : la valeur de valeur une fois mise a l'echelle

Définition à la ligne 147 du fichier autre.c.

Voici le graphe des appelants de cette fonction :

### 4.5.2.2 joue1000Partie()

```
void joue1000Partie (
            int nbPartie )
```

joue n partie aves uniquement des ia pour controler si tout ce passe bien et reuceillir des statistique

**Paramètres**

| | |
|---|---|
| *int* | nbPartie nombre de partie que l'on veut jouer |

**Renvoie**

voir

Définition à la ligne 153 du fichier autre.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.5.2.3 joueurSuivant()

```
Joueur joueurSuivant (
            Joueur joueur )
```

passe au joueur suivant

**Paramètres**

| | |
|---|---|
| *Joueur* | joueur : joueur actuel |

**Renvoie**

Joueur : le joueur suivant

Définition à la ligne 11 du fichier autre.c.

Voici le graphe des appelants de cette fonction :

#### 4.5.2.4 nbAleatoire()

```
int nbAleatoire (
            int mini,
            int maxi )
```

renvoit in nombre aléatoire entre deux bornes

**Paramètres**

| | |
|---|---|
| *int* | mini : borne inférieur |
| *int* | maxi : borne surpérieur |

**Renvoie**

int : le nombre aléatoire

Définition à la ligne 21 du fichier autre.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.5.2.5 pointPli()

```
int pointPli (
            Carte pli[],
            Couleur atout,
            int nbCarte )
```

compte le nombre de points d'un pli

**Paramètres**

| | |
|---|---|
| *Carte* | pli []: tableau des cartes possées dont on doit faire la somme des points |
| *Couleur* | atout: couleur de l'atout joué dans la manche |
| *nbCarte* | nombre de carte à compter |

**Renvoie**

points

Définition à la ligne 34 du fichier autre.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.5.2.6 setContrat()

```
void setContrat (
            Contrat * contrat,
            Joueur preneur,
            int nbPoint,
            Couleur atout,
            Coinche coinche )
```

définit un contrat

**Paramètres**

| | |
|---|---|
| *Contat* | ∗contrat : pointeur sur le contrat a modifier |
| *Joueur* | preneur : preneur du contrat |
| *int* | nbPoint : nombre de point |
| *Couleur* | atout : couleur de l'aout |
| *Coinche* | coinche : normal, coinché , surcoinché |

**Renvoie**

void

Définition à la ligne 26 du fichier autre.c.

Voici le graphe des appelants de cette fonction :

## 4.6 Référence du fichier autre.h

ficher header contenant les prototype des fonctions qui n'allait dans aucun autre fichier

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

## Fonctions

- Joueur joueurSuivant (Joueur joueur)
  *passe au joueur suivant*
- int nbAleatoire (int mini, int maxi)
  *renvoit in nombre aléatoire entre deux bornes*
- void setContrat (Contrat *contrat, Joueur preneur, int nbPoint, Couleur atout, Coinche coinche)
  *définit un contrat*
- int pointPli (Carte pli[ ], Couleur atout, int nbCarte)
  *compte le nombre de points d'un pli*
- float ajusteEchelle (float valeur, float entreMin, float entreMax, float sortieMin, float sortieMax)
  *met a l'echelle la valeur d'une variable comprise entre deux bornes pour que la variable de retour soit compris entre deux autres bornes*
- void joue1000Partie (int nbPartie)
  *joue n partie aves uniquement des ia pour controler si tout ce passe bien et reuceillir des statistique*

### 4.6.1 Description détaillée

ficher header contenant les prototype des fonctions qui n'allait dans aucun autre fichier

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.6.2 Documentation des fonctions

#### 4.6.2.1 ajusteEchelle()

```
float ajusteEchelle (
            float valeur,
            float entreMin,
            float entreMax,
            float sortieMin,
            float sortieMax )
```

met a l'echelle la valeur d'une variable comprise entre deux bornes pour que la variable de retour soit compris entre deux autres bornes

**Paramètres**

| | |
|---|---|
| *float* | valeur : valeur a mettre à l'echelle |
| *float* | entreMin : valeur minimum que peut prendre l'entrée |
| *float* | entreMax : valeur maximum que peut prendre l'entrée |
| *float* | sortieMin : valeur minimum que peut prendre la sortie |
| *float* | sortieMax : valeur maximum que peut prendre la sortie |

**Renvoie**

float : la valeur de valeur une fois mise a l'echelle

Définition à la ligne 147 du fichier autre.c.

Voici le graphe des appelants de cette fonction :

#### 4.6.2.2 joue1000Partie()

```
void joue1000Partie (
            int nbPartie )
```

joue n partie aves uniquement des ia pour controler si tout ce passe bien et reuceillir des statistique

**Paramètres**

| | |
|---|---|
| *int* | nbPartie nombre de partie que l'on veut jouer |

**Renvoie**

voir

Définition à la ligne 153 du fichier autre.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.6.2.3 joueurSuivant()

```
Joueur joueurSuivant (
            Joueur joueur )
```

passe au joueur suivant

**Paramètres**

| | |
|---|---|
| *Joueur* | joueur : joueur actuel |

**Renvoie**

Joueur : le joueur suivant

Définition à la ligne 11 du fichier autre.c.

Voici le graphe des appelants de cette fonction :

#### 4.6.2.4 nbAleatoire()

```
int nbAleatoire (
            int mini,
            int maxi )
```

renvoit in nombre aléatoire entre deux bornes

**Paramètres**

| | |
|---|---|
| *int* | mini : borne inférieur |
| *int* | maxi : borne surpérieur |

**Renvoie**

int : le nombre aléatoire

Définition à la ligne 21 du fichier autre.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.6.2.5 pointPli()

```
int pointPli (
            Carte pli[],
            Couleur atout,
            int nbCarte )
```

compte le nombre de points d'un pli

**Paramètres**

| | |
|---|---|
| *Carte* | pli []: tableau des cartes possées dont on doit faire la somme des points |
| *Couleur* | atout: couleur de l'atout joué dans la manche |
| *nbCarte* | nombre de carte à compter |

**Renvoie**

points

Définition à la ligne 34 du fichier autre.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.6.2.6 setContrat()

```
void setContrat (
            Contrat * contrat,
            Joueur preneur,
            int nbPoint,
            Couleur atout,
            Coinche coinche )
```

définit un contrat

**Paramètres**

| | |
|---|---|
| *Contat* | ∗contrat : pointeur sur le contrat a modifier |
| *Joueur* | preneur : preneur du contrat |
| *int* | nbPoint : nombre de point |
| *Couleur* | atout : couleur de l'aout |
| *Coinche* | coinche : normal, coinché , surcoinché |

**Renvoie**

void

Définition à la ligne 26 du fichier autre.c.

Voici le graphe des appelants de cette fonction :

## 4.7 Référence du fichier formatageChaine.c

ficher source contenant les fonctions relative au formatage

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

Graphe des dépendances par inclusion de formatageChaine.c:

### Fonctions

- int centreChaine (char chaineInitial[ ], char chaineFinal[ ], int longeurChaine)
  *transphorme une chaine de caractère de longeur inconue en une chaine de caratère fixé de tel sorte que le texte soit centré*
- int centreModifieChaine (char chaine[ ], int longeurChaine)
  *transforme une chaine de cararactère en sa vertion centrée*
- int decoupeChaine (char chaineInitiale[ ], char ∗chaineFinale, int tailleLigne, int nbLigne)
  *fontion qui permet de découper une chaine de caracère en plusieur chaine de taille fixée en coupant sur un espace*
- void rempliEspace (char ∗chaine, int nbEspace)
  *rempli une chaine de caractère de nbEspace caractère espace ' '*
- void formateCarte (Carte carte[ ], char ∗chaineFinale, int nbCarte, int tailleChaine, int version)
  *formate les chaine de caracère qui affiche la valeur et la couleur des cartes*
- void stockeInfoCarte (Carte carte, char ∗valeur, char ∗couleur, int version, int tailleChaine)
  *met dans une chaine de caractère la valeur et la couleur d'une carte*
- void formatePseudo (Joueur joueur, char ∗pseudo[ ], int tailleChaine, char chaineFinal[ ], int version)
  *stocke dans une chaine de caractère le pseudo d'un des joueur*
- void formateContrat (Contrat contrat, char ∗chaineFinal, int tailleLigne, char ∗pseudo[ ])
  *stocke dans des chaine de caractère les info du contrat*
- int aligneModifieChaine (char chaine[ ], int longeurChaine)
  *transforme une chaine de cararactère en sa vertion aligné a gauche*
- void genereMessage (char message[ ], Joueur parle, char ∗pseudo[ ], Carte carteJoue, int score, TypeMessage typeMessage)
  *génere un message et l'enregistre dans une chaine de caractère*

### 4.7.1 Description détaillée

ficher source contenant les fonctions relative au formatage

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.7.2 Documentation des fonctions

#### 4.7.2.1 aligneModifieChaine()

```
int aligneModifieChaine (
            char chaine[],
            int longeurChaine )
```

transforme une chaine de cararactère en sa vertion aligné a gauche

**Paramètres**

| | |
|---|---|
| *char* | chaine : chaine a modifier |
| *int* | longeurChaine |

**Renvoie**

int : 0 si tout c'est bien passé

Définition à la ligne 299 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.7.2.2 centreChaine()

```
int centreChaine (
            char chaineInitial[],
            char chaineFinal[],
            int longeurChaine )
```

transphorme une chaine de caractère de longeur inconue en une chaine de caratère fixé de tel sorte que le texte soit centré

**Paramètres**

| | |
|---|---|
| *char* | chaineInitial : chaine a modifier |
| *char* | chaineFinal : pointeur vers la variable qui stocke la chainne final |
| *int* | longeurChaine : longeur de la chaine final |

**Renvoie**

int : 0 si tout va bien, 1 si la chaine est Initiale test plus longue que la longeur désirée

Définition à la ligne 10 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.7.2.3 centreModifieChaine()

```
int centreModifieChaine (
            char chaine[],
            int longeurChaine )
```

transforme une chaine de cararactère en sa vertion centrée

**Paramètres**

| | |
|---|---|
| *char* | chaine : chaine a modifier |
| *int* | longeurChaine |

**Renvoie**

int : 0 si tout c'est bien passé

Définition à la ligne 35 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.7.2.4 decoupeChaine()

```
int decoupeChaine (
            char chaineInitiale[],
            char * chaineFinale,
            int tailleLigne,
            int nbLigne )
```

fontion qui permet de découper une chaine de caracère en plusieur chaine de taille fixée en coupant sur un espace

**Paramètres**

| | |
|---|---|
| *chaineInitiale[ ]* | : chaine a decouper |
| *chaineFinale[ ][ ]* | : tableau contenant les chaine un fois découpé |
| *tailleLigne* | : nobre de caracète maximun par ligne |
| *nbLinge* | : nombre maximun de ligne |

**Renvoie**

0 si tout va bien 1 si la chaine est trop longue

Définition à la ligne 45 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.7.2.5 formateCarte()

```
void formateCarte (
            Carte carte[],
```

```
            char * chaineFinale,
            int nbCarte,
            int tailleChaine,
            int version )
```

formate les chaine de caracère qui affiche la valeur et la couleur des cartes

**Paramètres**

| | |
|---|---|
| *chaineFinale[ ][ ][ ]* | tableau a trois dimention qui stocke les chaine de caracère de la valeur et la couleurs des cartes |
| *carte[ ]* | : tableau de typpe Carte contenant les carte a afficher |
| *nbCarte* | : nombre de carte dans le tableau carte[] |
| *tailleChaine* | : taille des chaineFinale |
| *version* | : 0 si c'est la verion courte et 1 pour la verision longue |

**Renvoie**

void

Définition à la ligne 101 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.7.2.6 formateContrat()

```
void formateContrat (
            Contrat contrat,
            char * chaineFinal,
            int tailleLigne,
            char * pseudo[] )
```

stocke dans des chaine de caractère les info du contrat

**Paramètres**

| | |
|---|---|
| *Contrat* | contrat : contrat a afficher |
| *char* | ∗chainefinal : chaine final ou on enregistre les info du contrat |
| *int* | tailleLigne : taille d'une ligne |
| *char* | ∗pseudo[] : tableau qui contient les pseudo des joueur |

**Renvoie**

void

Définition à la ligne 251 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.7.2.7 formatePseudo()

```
void formatePseudo (
            Joueur joueur,
            char * pseudo[],
            int tailleChaine,
            char chaineFinal[],
            int version )
```

stocke dans une chaine de caractère le pseudo d'un des joueur

**Paramètres**

| | |
|---|---|
| *Joueur* | joueur : joueur dont on veut afficher le pseudo |
| *char* | ∗pseudo[] : tableau contennant les pseudo des joueur |
| *int* | tailleChaine : taille de la chaine final |
| *char* | chaineFinal[] : chaine final ou est stocké le pseudo |
| *int* | varsion : 1 pour la version ou la chaine finale est centrée 0 pour l'aligné a gauche |

**Renvoie**

void

Définition à la ligne 220 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.7.2.8 genereMessage()

```
void genereMessage (
            char message[],
            Joueur parle,
            char * pseudo[],
            Carte carteJoue,
            int score,
            TypeMessage typeMessage )
```

génere un message et l'enregistre dans une chaine de caractère

**Paramètres**

| | |
|---|---|
| *char* | message[] : tableau de char ou on enregistre le message |
| *Joueur* | parle : joueur qui pose une carte |
| *char* | ∗pseudo[] : tableau qui contient les pseudo des joueur |
| *Carte* | carteJoue : carte qui vine d'être jouée |
| *int* | score : score a afficher |
| *TypeMessage* | typeMessage : permet de savoir quelle message on veux afficher |

**Renvoie**

void

Définition à la ligne 314 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.7.2.9 rempliEspace()

```
void rempliEspace (
            char * chaine,
            int nbEspace )
```

rempli une chaine de caractère de nbEspace caractère espace ' '

**Paramètres**

| | |
|---|---|
| *chaine[ ]* | : chaine a remplir; |
| *nbEspace[ ]* | : nombre d'espaca a placer dans la chaine |

**Renvoie**

void

Définition à la ligne 93 du fichier formatageChaine.c.

Voici le graphe des appelants de cette fonction :

#### 4.7.2.10 stockeInfoCarte()

```
void stockeInfoCarte (
            Carte carte,
            char * valeur,
            char * couleur,
            int version,
            int tailleChaine )
```

met dans une chaine de caractère la valeur et la couleur d'une carte

**Paramètres**

| | |
|---|---|
| *carte* | : variable de type carte qui contient la carte a afficher |
| *valeur* | : pointeur vers la chaine qui stocke la valeur (ou la valeur et la couleur si mode court) |
| *couleur* | : pointeur vers la chaine qui stocke la couleur |
| *vesion* | : 0 si c'est la version courte et 1 pour la version longue |
| *tailleChaine* | : taille de la chaine de caratère dans laquel on écrit |

**Renvoie**

void

Définition à la ligne 108 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

# 4.8 Référence du fichier formatageChaine.h

ficher header contenant les prototypes des fonctions relative au formatage

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

## Fonctions

- int centreChaine (char chaineInitial[ ], char chaineFinal[ ], int longeurChaine)
  *transphorme une chaine de caractère de longeur inconue en une chaine de caratère fixé de tel sorte que le texte soit centré*
- int centreModifieChaine (char chaine[ ], int longeurChaine)
  *transforme une chaine de cararactère en sa vertion centrée*
- int decoupeChaine (char chaineInitiale[ ], char ∗chaineFinale, int tailleLigne, int nbLigne)
  *fontion qui permet de découper une chaine de caracère en plusieur chaine de taille fixée en coupant sur un espace*
- void rempliEspace (char ∗chaine, int nbEspace)
  *rempli une chaine de caractère de nbEspace caractère espace ' '*
- void formateCarte (Carte carte[ ], char ∗chaineFinale, int nbCarte, int tailleChaine, int version)
  *formate les chaine de caracère qui affiche la valeur et la couleur des cartes*
- void stockeInfoCarte (Carte carte, char ∗valeur, char ∗couleur, int version, int tailleChaine)
  *met dans une chaine de caractère la valeur et la couleur d'une carte*
- void formatePseudo (Joueur joueur, char ∗pseudo[ ], int tailleChaine, char chaineFinal[ ], int version)
  *stocke dans une chaine de caractère le pseudo d'un des joueur*
- void formateContrat (Contrat contrat, char ∗chaineFinal, int tailleLigne, char ∗pseudo[ ])
  *stocke dans des chaine de caractère les info du contrat*
- int aligneModifieChaine (char chaine[ ], int longeurChaine)
  *transforme une chaine de cararactère en sa vertion aligné a gauche*
- void genereMessage (char message[ ], Joueur parle, char ∗pseudo[ ], Carte carteJoue, int score, TypeMessage typeMessage)
  *génere un message et l'enregistre dans une chaine de caractère*

### 4.8.1 Description détaillée

ficher header contenant les prototypes des fonctions relative au formatage

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.8.2 Documentation des fonctions

#### 4.8.2.1 aligneModifieChaine()

```
int aligneModifieChaine (
            char chaine[],
            int longeurChaine )
```

transforme une chaine de cararactère en sa vertion aligné a gauche

**Paramètres**

| | |
|---|---|
| *char* | chaine : chaine a modifier |
| *int* | longeurChaine |

**Renvoie**

int : 0 si tout c'est bien passé

Définition à la ligne 299 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.8.2.2 centreChaine()

```
int centreChaine (
            char chaineInitial[],
            char chaineFinal[],
            int longeurChaine )
```

transphorme une chaine de caractère de longeur inconue en une chaine de caratère fixé de tel sorte que le texte soit centré

**Paramètres**

| | |
|---|---|
| *char* | chaineInitial : chaine a modifier |
| *char* | chaineFinal : pointeur vers la variable qui stocke la chainne final |
| *int* | longeurChaine : longeur de la chaine final |

**Renvoie**

int : 0 si tout va bien, 1 si la chaine est Initiale test plus longue que la longeur désirée

Définition à la ligne 10 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.8.2.3 centreModifieChaine()

```
int centreModifieChaine (
            char chaine[],
            int longeurChaine )
```

transforme une chaine de cararactère en sa vertion centrée

**Paramètres**

| | |
|---|---|
| *char* | chaine : chaine a modifier |
| *int* | longeurChaine |

**Renvoie**

int : 0 si tout c'est bien passé

Définition à la ligne 35 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.8.2.4 decoupeChaine()

```
int decoupeChaine (
            char chaineInitiale[],
            char * chaineFinale,
            int tailleLigne,
            int nbLigne )
```

fontion qui permet de découper une chaine de caracère en plusieur chaine de taille fixée en coupant sur un espace

**Paramètres**

| | |
|---|---|
| *chaineInitiale[ ]* | : chaine a decouper |
| *chaineFinale[ ][ ]* | : tableau contenant les chaine un fois découpé |
| *tailleLigne* | : nobre de caracète maximun par ligne |
| *nbLinge* | : nombre maximun de ligne |

**Renvoie**

0 si tout va bien 1 si la chaine est trop longue

Définition à la ligne 45 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.8.2.5 formateCarte()

```
void formateCarte (
            Carte carte[],
            char * chaineFinale,
            int nbCarte,
            int tailleChaine,
            int version )
```

formate les chaine de caracère qui affiche la valeur et la couleur des cartes

**Paramètres**

| | |
|---|---|
| *chaineFinale[ ][ ][ ]* | tableau a trois dimention qui stocke les chaine de caracère de la valeur et la couleurs des cartes |
| *carte[ ]* | : tableau de typpe Carte contenant les carte a afficher |
| *nbCarte* | : nombre de carte dans le tableau carte[] |
| *tailleChaine* | : taille des chaineFinale |
| *version* | : 0 si c'est la verion courte et 1 pour la verision longue |

**Renvoie**

void

Définition à la ligne 101 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.8.2.6 formateContrat()

```
void formateContrat (
            Contrat contrat,
            char * chaineFinal,
            int tailleLigne,
            char * pseudo[] )
```

stocke dans des chaine de caractère les info du contrat

**Paramètres**

| | |
|---|---|
| *Contrat* | contrat : contrat a afficher |
| *char* | ∗chainefinal : chaine final ou on enregistre les info du contrat |
| *int* | tailleLigne : taille d'une ligne |
| *char* | ∗pseudo[] : tableau qui contient les pseudo des joueur |

**Renvoie**

void

Définition à la ligne 251 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.8.2.7 formatePseudo()

```
void formatePseudo (
            Joueur joueur,
            char * pseudo[],
            int tailleChaine,
            char chaineFinal[],
            int version )
```

stocke dans une chaine de caractère le pseudo d'un des joueur

**Paramètres**

| | |
|---|---|
| *Joueur* | joueur : joueur dont on veut afficher le pseudo |
| *char* | ∗pseudo[] : tableau contennant les pseudo des joueur |
| *int* | tailleChaine : taille de la chaine final |
| *char* | chaineFinal[] : chaine final ou est stocké le pseudo |
| *int* | varsion : 1 pour la version ou la chaine finale est centrée 0 pour l'aligné a gauche |

**Renvoie**

void

Définition à la ligne 220 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.8.2.8 genereMessage()

```
void genereMessage (
            char message[],
            Joueur parle,
            char * pseudo[],
            Carte carteJoue,
            int score,
            TypeMessage typeMessage )
```

génere un message et l'enregistre dans une chaine de caractère

**Paramètres**

| | |
|---|---|
| *char* | message[] : tableau de char ou on enregistre le message |
| *Joueur* | parle : joueur qui pose une carte |
| *char* | ∗pseudo[] : tableau qui contient les pseudo des joueur |
| *Carte* | carteJoue : carte qui vine d'être jouée |
| *int* | score : score a afficher |
| *TypeMessage* | typeMessage : permet de savoir quelle message on veux afficher |

**Renvoie**

void

Définition à la ligne 314 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.8.2.9 rempliEspace()

```
void rempliEspace (
            char * chaine,
            int nbEspace )
```

rempli une chaine de caractère de nbEspace caractère espace ' '

**Paramètres**

| | |
|---|---|
| *chaine[ ]* | : chaine a remplir; |
| *nbEspace[ ]* | : nombre d'espaca a placer dans la chaine |

**Renvoie**

void

Définition à la ligne 93 du fichier formatageChaine.c.

Voici le graphe des appelants de cette fonction :

#### 4.8.2.10 stockeInfoCarte()

```
void stockeInfoCarte (
            Carte carte,
            char * valeur,
            char * couleur,
            int version,
            int tailleChaine )
```

met dans une chaine de caractère la valeur et la couleur d'une carte

**Paramètres**

| | |
|---|---|
| *carte* | : variable de type carte qui contient la carte a afficher |
| *valeur* | : pointeur vers la chaine qui stocke la valeur (ou la valeur et la couleur si mode court) |
| *couleur* | : pointeur vers la chaine qui stocke la couleur |
| *vesion* | : 0 si c'est la version courte et 1 pour la version longue |
| *tailleChaine* | : taille de la chaine de caratère dans laquel on écrit |

**Renvoie**

void

Définition à la ligne 108 du fichier formatageChaine.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

## 4.9 Référence du fichier general.c

ficher source contenant les fonctions les plus impotantes pour le jeu de la belote

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

Graphe des dépendances par inclusion de general.c:

### Fonctions

- void initialisation (int nbLigneFenetre, int nbColloneFenetre)
  *fonction qui gÃ¨re l'initialitation et le lancement du programme*
- void menuPrincipal ()
  *fonction qui gÃ¨re le debut de la partie*
- int nouvellePartie (char ∗pseudo[ ], Joueur utilisateur, int ∗pStatistique, int infoEcritureFicher[ ])
  *lancement d'une nouvelle partie*
- char manche (char ∗pseudo[ ], int score[ ], Joueur dealer, Joueur utilisateur, int ∗pStatistique)
  *lancement d'une manche*

- Contrat annonceContrat (char ∗pseudo[ ], Joueur dealer, Carte ∗pCarteMain, Joueur utilisateur)
  *gÃ¨re l'annonce et la surenchÃ¨re des contats par les joueurs*
- Contrat proposeContrat (Contrat dernierContrat, Joueur parle, char ∗pseudo[ ], Carte ∗pCarteMain, Joueur utilisateur)
  *proposition d'un contrat par un joueur*
- Joueur pli (Contrat contrat, Joueur premierAJouer, char ∗pseudo[ ], Carte ∗pCarteMain, int pointManche[ ], int pointAnonce[ ], char belote[ ], Carte cartePli[ ], Carte carteAncienPli[ ], int score[ ], Joueur utilisateur, int numPli)
  *fonction qui gÃ¨re un pli*
- int poseCarte (Joueur joueur, int numCarte, Carte ∗pMainJoueurs, Carte pli[ ], int carteRestante)
  *fonction qui verifie la validitÃ© d'une carte est la pose*
- char calculPointManche (Contrat contrat, int pointManche[ ], int pointAnonce[ ], int pointBelote[ ], int score[ ], char ∗pseudo[ ], Joueur utilisateur)
  *compte les point gagnÃ© par chaque joueur au cour de la manche e fonction du contrat et ajoute ces point au tableau qui contient tout les score de la partie*

## 4.9.1 Description détaillée

ficher source contenant les fonctions les plus impotantes pour le jeu de la belote

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

## 4.9.2 Documentation des fonctions

### 4.9.2.1 annonceContrat()

```
Contrat annonceContrat (
            char * pseudo[],
            Joueur dealer,
            Carte * pCarteMain,
            Joueur utilisateur )
```

gÃ¨re l'annonce et la surenchÃ¨re des contats par les joueurs

**Paramètres**

| | |
|---|---|
| *char* | ∗pseudo[]: tableau de pointeurs contenant les pseudos des diffÃ©rents joueurs |
| *Joueur* | dealer : joueur qui distibue les cartes |
| *Carte* | ∗pCarteMain : pointeur sur le tableau qui stocke les cartes dans la main de chaque joueur |
| *Joueuer* | utilisateur : donne la position de l'utilisateur, mettre SANS_Joueur pour faire une partie avec uniquement des ordinateur |

**Renvoie**

Contrat : contrat final qui a ete choisi pour la partie

< Affichage du choix du joueur

Définition à la ligne 287 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.9.2.2 calculPointManche()

```
char calculPointManche (
            Contrat contrat,
            int pointManche[],
            int pointAnonce[],
            int pointBelote[],
            int score[],
            char * pseudo[],
            Joueur utilisateur )
```

compte les point gagnÃ© par chaque joueur au cour de la manche e fonction du contrat et ajoute ces point au tableau qui contient tout les score de la partie

**Paramètres**

| | |
|---|---|
| *Contrat* | contrat : contrat qui a Ã©tÃ© choisi durant la manche |
| *int* | pointManche[] : tableau qui contient les points de la manche |
| *int* | pointAnonce[] : tableau qui contient les points d'anonce de la manche |
| *int* | pointBelote[] : tableau contenant les point de belote rebelote de chaque joueur |
| *int* | score[] : tableau qui contient les score de la partie |
| *char* | ∗pseudo[]: tableau de pointeur contenant les pseudo des diffÃ©rents joueur |
| *Joueur* | utilisateur : donne la position de l'utilisateur, mettre SANS_Joueur pour faire une partie avec uniquement des ordinateur |

**Renvoie**

char : 1 si le contrtat a Ã©tÃ© rÃ©ussi et 0 si le contrat a Ã©chouÃ©

< On reche si le contrat est rampli

< cas ou un des joueur a pris un capot

< cas ou un des joueur a pris une gÃ©nÃ©rale

< Le contrat est rempli si le les point de l'Ã©quipe est suppÃ©rieur a ce qui anoncÃ© et si il on 82 point sans les anonces

< on affiche ce qui c'est passÃ©

Définition à la ligne 476 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.9.2.3 initialisation()

```
void initialisation (
            int nbLigneFenetre,
            int nbColloneFenetre )
```

fonction qui gÃ¨re l'initialitation et le lancement du programme

**Paramètres**

| | |
|---|---|
| *nbLigneFenetre* | : taille verticale de la fenÃªtre |
| *nbColloneFenetre* | : taille horizonotale de la fenÃªtre |

**Renvoie**

void

Définition à la ligne 11 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.9.2.4 manche()

```
char manche (
            char * pseudo[],
            int score[],
            Joueur dealer,
            Joueur utilisateur,
            int * pStatistique )
```

lancement d'une manche

**Paramètres**

| | |
|---|---|
| *char* | ∗pseudo[] : tableau de pointeurs contenant les pseudos des diffÃ©rents joueurs |
| *int* | score[] : tableau contenant les scores des joueur |
| *Joueur* | dealer : joueur qui distibue les cartes |
| *Joueuer* | utilisateur : donne la position de l'utilisateur, mettre SANS_Joueur pour faire une partie avec uniquement des ordinateur |
| *int* | ∗pStatistique : pointeur sur un tableau 4∗4 qui contient des statistique sur chaque joueur |

**Renvoie**

char : 0 si tout le monde passe, 1 si la manche a bien lieu

< tableau qui stocke les anonce de chaque joueur, on y ajoute 1 pour le roi d'atout puis 1 pour la dame d'atout

< distribution des cartes

< On cherche si les joueurs on des anonce a faire

< On passe a la phase suivante uniquement si un contrat a ete pris sinon on relance une manche

< si un joueur a fait une belote rebelote il gagne 20 point

Définition à la ligne 188 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.9.2.5 menuPrincipal()

```
void menuPrincipal ( )
```

fonction qui gÃ¨re le debut de la partie

**Renvoie**

void

< contrÃ´le d'acquisition avec rÃ©affichage de l'interfface

<executer la fonction nouvelle partie

Définition à la ligne 16 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.9.2.6 nouvellePartie()

```
int nouvellePartie (
            char * pseudo[],
            Joueur utilisateur,
            int * pStatistique,
            int infoEcritureFicher[] )
```

lancement d'une nouvelle partie

**Paramètres**

| | |
|---|---|
| *char* | ∗pseudo[] : tableau de pointeurs contenant les pseudos des diffÃ©rents joueurs |
| *Joueuer* | utilisateur : donne la position de l'utilisateur, mettre SANS_Joueur pour faire une partie avec uniquement des ordinateur |
| *int* | ∗pStatistique : pointeur sur un tableau 4∗4 qui contient des statistique sur chaque joueur |
| *int* | infoEcritureFichier[] : tableau contenant les info qui devront etre Ã©crite dans le fichier |

**Renvoie**

int : nombre de manche jusqu'a la victoir d'une Ã©quipe

< On change l'affichage pour avoir un affichage plus simple a lire lors d'une partie entre 4 ordinateur

< a suprimer une fois de debug fini

< Fin de partie

< affichage des resultat

< L'utilisatuer et nord gagnent

< est et touest gagnent

Définition à la ligne 99 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.9.2.7 pli()

```
Joueur pli (
            Contrat contrat,
            Joueur premierAJouer,
            char * pseudo[],
            Carte * pCarteMain,
            int pointManche[],
            int pointAnonce[],
            char belote[],
            Carte cartePli[],
            Carte carteAncienPli[],
            int score[],
            Joueur utilisateur,
            int numPli )
```

fonction qui gÃ¨re un pli

**Paramètres**

| | |
|---|---|
| *Contrat* | contrat : contrat qui a Ã©tÃ© choisipour cette manche |
| *Joueur* | premierAjouer : premier joueur a jouer dans le pli |
| *char* | ∗pseudo[]: tableau de pointeur contenant les pseudo des diffÃ©rents joueur |
| *Carte* | ∗pCarteMain : pointeur sur le tableau qui stocke les carte dans la main de chaque joueur |
| *int* | pointManche[] : tableau contenant les point de chaque joueur dans la manche |
| *int* | pointAnonce[] : tableau contenant les point d'anonce de chaque joueur |
| *char* | belote[] : tableau contenant les ancone belote rebelote de chaque joueur |
| *Carte* | cartePli[] : tableau contenant les 4 carte du pli |
| *Carte* | carteAncienPli[] : tableau contenant les carte du pli prÃ©cÃ©dent |
| *int* | score[] : tableau qui contient les score de la partie |
| *Joueur* | utilisateur : donne la position de l'utilisateur, mettre SANS_Joueur pour faire une partie avec uniquement des ordinateur |

**Renvoie**

Joueur : le vainceur du pli

< interface de pli Utilisateur

< interface de pli ordinateur

< on regarde si une dame ou un roi d'atout vien d'etre jouer pour ajouter au tableau belote[]

< affichage de la carte qui vien d'etre jouÃ©e

< 10 de der, on ajoute 10 point au vainceur si on est dans le dernier pli

Définition à la ligne 390 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.9.2.8 poseCarte()

```
int poseCarte (
            Joueur joueur,
            int numCarte,
            Carte * pMainJoueurs,
            Carte pli[],
            int carteRestante )
```

fonction qui verifie la validitÃ© d'une carte est la pose

**Paramètres**

| | |
|---|---|
| *joueur* | variable qui defini la position du joueur qui effectue l'action |
| *numCarte* | variable qui defini la carte choisie par le joueur |
| *pMainJoueurs* | : pointeur qui renvoie vers le tableau de la main de tous les joueurs |
| *pPli* | : pointeur qui renvoie vers le tableau des carte jouÃ©es dans le pli |

**Renvoie**

Définition à la ligne 463 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.9.2.9 proposeContrat()

```
Contrat proposeContrat (
            Contrat dernierContrat,
            Joueur parle,
            char * pseudo[],
            Carte * pCarteMain,
            Joueur utilisateur )
```

proposition d'un contrat par un joueur

**Paramètres**

| | |
|---|---|
| *Contrat* | dernierContrat : dernier contrat proposÃ© |
| *Joueur* | parle : joueur qui parle |
| *char* | ∗pseudo[]: tableau de pointeurs contenant les pseudos des diffÃ©rents joueurs |
| *Carte* | ∗pCarteMain : pointeur sur le tableau qui stocke les cartes dans la main de chaque joueurs |
| *Joueur* | utilisateur : donne la position de l'utilisateur, mettre SANS_Joueur pour faire une partie avec |
| | uniquement des ordinateur |

**Renvoie**

Contrat : nouveau contrat proposÃ© par le joueur

< acquisition par l'utilisateur

< choix par l'ia d'un contrat

Définition à la ligne 371 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

## 4.10 Référence du fichier general.h

ficher header contenant les prototypes des fonctions les plus impotantes pour le jeu de la belote

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

### Fonctions

- void initialisation (int nbLigneFenetre, int nbColloneFenetre)
  *fonction qui gÃ¨re l'initialitation et le lancement du programme*
- void menuPrincipal ()
  *fonction qui gÃ¨re le debut de la partie*
- int nouvellePartie (char ∗pseudo[ ], Joueur utilisateur, int ∗pStatistique, int infoEcritureFicher[ ])
  *lancement d'une nouvelle partie*
- char manche (char ∗pseudo[ ], int score[ ], Joueur dealer, Joueur utilisateur, int ∗pStatistique)
  *lancement d'une manche*
- Contrat annonceContrat (char ∗pseudo[ ], Joueur dealer, Carte ∗pCarteMain, Joueur utilisateur)
  *gÃ¨re l'annonce et la surenchÃ¨re des contats par les joueurs*
- Contrat proposeContrat (Contrat dernierContrat, Joueur parle, char ∗pseudo[ ], Carte ∗pCarteMain, Joueur utilisateur)
  *proposition d'un contrat par un joueur*
- Joueur pli (Contrat contrat, Joueur premierAJouer, char ∗pseudo[ ], Carte ∗pCarteMain, int pointManche[ ], int pointAnonce[ ], char belote[ ], Carte cartePli[ ], Carte carteAncienPli[ ], int score[ ], Joueur utilisateur, int numPli)
  *fonction qui gÃ¨re un pli*
- int poseCarte (Joueur joueur, int numCarte, Carte ∗pMainJoueurs, Carte pli[ ], int carteRestante)
  *fonction qui verifie la validitÃ© d'une carte est la pose*
- char calculPointManche (Contrat contrat, int pointManche[ ], int pointAnonce[ ], int pointBelote[ ], int score[ ], char ∗pseudo[ ], Joueur utilisateur)
  *compte les point gagnÃ© par chaque joueur au cour de la manche e fonction du contrat et ajoute ces point au tableau qui contient tout les score de la partie*

### 4.10.1 Description détaillée

ficher header contenant les prototypes des fonctions les plus impotantes pour le jeu de la belote

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.10.2 Documentation des fonctions

#### 4.10.2.1 annonceContrat()

```
Contrat annonceContrat (
            char * pseudo[],
            Joueur dealer,
            Carte * pCarteMain,
            Joueur utilisateur )
```

gÃ¨re l'annonce et la surenchÃ¨re des contats par les joueurs

**Paramètres**

| | |
|---|---|
| *char* | ∗pseudo[]: tableau de pointeurs contenant les pseudos des diffÃ©rents joueurs |
| *Joueur* | dealer : joueur qui distibue les cartes |
| *Carte* | ∗pCarteMain : pointeur sur le tableau qui stocke les cartes dans la main de chaque joueur |
| *Joueuer* | utilisateur : donne la position de l'utilisateur, mettre SANS_Joueur pour faire une partie avec uniquement des ordinateur |

**Renvoie**

Contrat : contrat final qui a ete choisi pour la partie

< Affichage du choix du joueur

Définition à la ligne 287 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.10.2.2 calculPointManche()

```
char calculPointManche (
            Contrat contrat,
            int pointManche[],
            int pointAnonce[],
            int pointBelote[],
            int score[],
            char * pseudo[],
            Joueur utilisateur )
```

compte les point gagnÃ© par chaque joueur au cour de la manche e fonction du contrat et ajoute ces point au tableau qui contient tout les score de la partie

**Paramètres**

| | |
|---|---|
| *Contrat* | contrat : contrat qui a Ã©tÃ© choisi durant la manche |
| *int* | pointManche[] : tableau qui contient les points de la manche |
| *int* | pointAnonce[] : tableau qui contient les points d'anonce de la manche |
| *int* | pointBelote[] : tableau contenant les point de belote rebelote de chaque joueur |
| *int* | score[] : tableau qui contient les score de la partie |
| *char* | ∗pseudo[]: tableau de pointeur contenant les pseudo des diffÃ©rents joueur |
| *Joueur* | utilisateur : donne la position de l'utilisateur, mettre SANS_Joueur pour faire une partie avec uniquement des ordinateur |

**Renvoie**

char : 1 si le contrtat a Ã©tÃ© rÃ©ussi et 0 si le contrat a Ã©chouÃ©

< On reche si le contrat est rampli

< cas ou un des joueur a pris un capot

< cas ou un des joueur a pris une gÃ©nÃ©rale

< Le contrat est rempli si le les point de l'Ã©quipe est suppÃ©rieur a ce qui anoncÃ© et si il on 82 point sans les anonces

< on affiche ce qui c'est passÃ©

Définition à la ligne 476 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.10.2.3 initialisation()

```
void initialisation (
            int nbLigneFenetre,
            int nbColloneFenetre )
```

fonction qui gÃ¨re l'initialitation et le lancement du programme

**Paramètres**

| | |
|---|---|
| *nbLigneFenetre* | : taille verticale de la fenÃªtre |
| *nbColloneFenetre* | : taille horizonotale de la fenÃªtre |

**Renvoie**

void

Définition à la ligne 11 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.10.2.4 manche()

```
char manche (
            char * pseudo[],
            int score[],
            Joueur dealer,
            Joueur utilisateur,
            int * pStatistique )
```

lancement d'une manche

**Paramètres**

| | |
|---|---|
| *char* | ∗pseudo[] : tableau de pointeurs contenant les pseudos des diffÃ©rents joueurs |
| *int* | score[] : tableau contenant les scores des joueur |
| *Joueur* | dealer : joueur qui distibue les cartes |
| *Joueuer* | utilisateur : donne la position de l'utilisateur, mettre SANS_Joueur pour faire une partie avec uniquement des ordinateur |
| *int* | ∗pStatistique : pointeur sur un tableau 4∗4 qui contient des statistique sur chaque joueur |

**Renvoie**

char : 0 si tout le monde passe, 1 si la manche a bien lieu

< tableau qui stocke les anonce de chaque joueur, on y ajoute 1 pour le roi d'atout puis 1 pour la dame d'atout

< distribution des cartes

< On cherche si les joueurs on des anonce a faire

< On passe a la phase suivante uniquement si un contrat a ete pris sinon on relance une manche

< si un joueur a fait une belote rebelote il gagne 20 point

Définition à la ligne 188 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.10.2.5 menuPrincipal()

```
void menuPrincipal ( )
```

fonction qui gÃ¨re le debut de la partie

**Renvoie**

void

< contrÃ´le d'acquisition avec rÃ©affichage de l'interfface

<executer la fonction nouvelle partie

Définition à la ligne 16 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.10.2.6 nouvellePartie()

```
int nouvellePartie (
            char * pseudo[],
            Joueur utilisateur,
            int * pStatistique,
            int infoEcritureFicher[] )
```

lancement d'une nouvelle partie

**Paramètres**

| | |
|---|---|
| *char* | ∗pseudo[] : tableau de pointeurs contenant les pseudos des diffÃ©rents joueurs |
| *Joueuer* | utilisateur : donne la position de l'utilisateur, mettre SANS_Joueur pour faire une partie avec uniquement des ordinateur |
| *int* | ∗pStatistique : pointeur sur un tableau 4∗4 qui contient des statistique sur chaque joueur |
| *int* | infoEcritureFichier[] : tableau contenant les info qui devront etre Ã©crite dans le fichier |

**Renvoie**

int : nombre de manche jusqu'a la victoir d'une Ã©quipe

< On change l'affichage pour avoir un affichage plus simple a lire lors d'une partie entre 4 ordinateur

< a suprimer une fois de debug fini

< Fin de partie

< affichage des resultat

< L'utilisatuer et nord gagnent

< est et touest gagnent

Définition à la ligne 99 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.10.2.7 pli()

```
Joueur pli (
            Contrat contrat,
            Joueur premierAJouer,
            char * pseudo[],
            Carte * pCarteMain,
            int pointManche[],
            int pointAnonce[],
            char belote[],
            Carte cartePli[],
            Carte carteAncienPli[],
            int score[],
            Joueur utilisateur,
            int numPli )
```

fonction qui gÃ¨re un pli

**Paramètres**

| | |
|---|---|
| *Contrat* | contrat : contrat qui a Ã©tÃ© choisipour cette manche |
| *Joueur* | premierAjouer : premier joueur a jouer dans le pli |
| *char* | ∗pseudo[]: tableau de pointeur contenant les pseudo des diffÃ©rents joueur |
| *Carte* | ∗pCarteMain : pointeur sur le tableau qui stocke les carte dans la main de chaque joueur |
| *int* | pointManche[] : tableau contenant les point de chaque joueur dans la manche |
| *int* | pointAnonce[] : tableau contenant les point d'anonce de chaque joueur |
| *char* | belote[] : tableau contenant les ancone belote rebelote de chaque joueur |
| *Carte* | cartePli[] : tableau contenant les 4 carte du pli |
| *Carte* | carteAncienPli[] : tableau contenant les carte du pli prÃ©cÃ©dent |
| *int* | score[] : tableau qui contient les score de la partie |

**Renvoie**

Joueur : le vainceur du pli

< interface de pli Utilisateur

< interface de pli ordinateur

< on regarde si une dame ou un roi d'atout vien d'etre jouer pour ajouter au tableau belote[]

< affichage de la carte qui vien d'etre jouÃ©e

< 10 de der, on ajoute 10 point au vainceur si on est dans le dernier pli

Définition à la ligne 390 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.10.2.8 poseCarte()

```
int poseCarte (
            Joueur joueur,
            int numCarte,
            Carte * pMainJoueurs,
            Carte pli[],
            int carteRestante )
```

fonction qui verifie la validitÃ© d'une carte est la pose

**Paramètres**

| | |
|---|---|
| *joueur* | variable qui defini la position du joueur qui effectue l'action |
| *numCarte* | variable qui defini la carte choisie par le joueur |
| *pMainJoueurs* | : pointeur qui renvoie vers le tableau de la main de tous les joueurs |
| *pPli* | : pointeur qui renvoie vers le tableau des carte jouÃ©es dans le pli |

**Renvoie**

Définition à la ligne 463 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.10.2.9 proposeContrat()

```
Contrat proposeContrat (
            Contrat dernierContrat,
            Joueur parle,
            char * pseudo[],
            Carte * pCarteMain,
            Joueur utilisateur )
```

proposition d'un contrat par un joueur

**Paramètres**

| | |
|---|---|
| *Contrat* | dernierContrat : dernier contrat proposÃ© |
| *Joueur* | parle : joueur qui parle |
| *char* | ∗pseudo[]: tableau de pointeurs contenant les pseudos des diffÃ©rents joueurs |
| *Carte* | ∗pCarteMain : pointeur sur le tableau qui stocke les cartes dans la main de chaque joueurs |
| *Joueur* | utilisateur : donne la position de l'utilisateur, mettre SANS_Joueur pour faire une partie avec uniquement des ordinateur |

**Renvoie**

Contrat : nouveau contrat proposÃ© par le joueur

< acquisition par l'utilisateur

< choix par l'ia d'un contrat

Définition à la ligne 371 du fichier general.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

## 4.11 Référence du fichier gestionCarte.c

ficher contenant les fonctions relative a la gestion des cartes

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

Graphe des dépendances par inclusion de gestionCarte.c:

### Fonctions

- void distribueCarte (Carte ∗pCarteMain)
  *distribue les cartes entre les différent joueur*
- void setCarte (Carte ∗carte, Valeur valeurCarte, Couleur couleurCarte)
  *donne une valeur et une couleur a une carte*
- void supprimeCarte (Carte carte[ ], int nbCarte, int carteASupprimer)
  *supprime une carte d'un tableau et réduit la taille de ce tableu par le bas*
- Joueur vainqueurPli (Carte pli[ ], Couleur atout, Joueur premierAJouer)
  *donne le vainquer d'un pli*
- float forceCarte (Carte carteACalculer, Couleur atout, Couleur entame)
  *calcul la force d'une carte, en faisant la probabilitée qu'elle a de gagné contre toutes les autres cartes*
- char carteValide (Carte cartePose, Carte pli[ ], Couleur atout, Carte ∗pCarteMainJoueur, Joueur premierA↩ Jouer, Joueur parle)
  *ddétermine si une carte peut etre posé par un joueur*
- char rechercherCarte (Carte ∗pCarte, int nbCarte, Couleur couleurCherche, Valeur valeurCherche)
  *cherche dans un tableau de carte si il y a une certaine couleur ou une certaine valeur ou les deux*
- char rechercherCarteSuperieur (Carte ∗pCarte, int nbCarte, Carte carteCherche, Couleur atout, Couleur entame)
  *cherche dans un tableau de carte si il y a une carte d'une force suppérieur a carteCherche*
- float sommeForceCarte (Carte ∗tableauCarte, int nbCarte, Couleur atout)
  *calucule la somme force des carte dans un tableau*
- void trieCarte (Carte tableauCarte[ ], int nbCarte, Couleur atout)
  *trie un tableau de cartes*
- char cartePlaceAvant (Carte carteRefference, Carte carteCompare, Couleur atout)
  *permet de savoir si une carte doit etre placée avant lors du trie des cartes*
- void rechercheAnnonce (Carte ∗pCarteMain, int pointAnonce[ ], Joueur utilisateur, char ∗pseudo[ ])
  *rechere dans un tableau de cartes qui correspond a la main d'un joureur si il peut faire des anonc, et si oui ajoute les points assacier au tableau dans le tableau des anonce*

### 4.11.1 Description détaillée

ficher contenant les fonctions relative a la gestion des cartes

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.11.2 Documentation des fonctions

#### 4.11.2.1 cartePlaceAvant()

```
char cartePlaceAvant (
            Carte carteRefference,
            Carte carteCompare,
            Couleur atout )
```

permet de savoir si une carte doit etre placée avant lors du trie des cartes

**Paramètres**

| | |
|---|---|
| *Carte* | carteRefference : carte par raport a laquelle on compare |
| *Carte* | carteCompare : carte que l'on compare |
| *Couleur* | atout : couleur de l'atout dans la manche |

**Renvoie**

char : renvoie 1 si la catre doit etre placée avant 0 sinon

< Les deux carte sont elle de la meme famille

< si carteCompate est un atout

< si la couleur de la carteCompare est devant celle de carteRefference dans l'Ã©numÃ©ration(ordre arbitraire)

Définition à la ligne 404 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.11.2.2 carteValide()

```
char carteValide (
            Carte cartePose,
            Carte pli[],
            Couleur atout,
            Carte * pCarteMainJoueur,
            Joueur premierAJouer,
            Joueur parle )
```

ddétermine si une carte peut etre posé par un joueur

**Paramètres**

| | |
|---|---|
| *Carte* | cartPose : carte que l'on veut poser |
| *Carte* | pli[] : tableau contenet les carte du pli |
| *Couleur* | atout : couleur de l'atout |
| *Carte* | ∗pCarteMainJoueur pointeur sur un tableau de 8 carte contenent les cartes dans la main du joueur |
| *Joueur* | premierAJouer : premier joueur a jouer dans le pli |
| *Joueur* | parle : joueur en train de jouer |

**Renvoie**

char 1 si la carte est valide 0 sinon

< Fonction faite a partir de l'oranigrame qui montre comment dÃ©terminer si une carte est valide a partir des rÃ¨gle

< On verifier que ce n'est pas une carte vide

< premiÃ¨re carte du plis ?

< Le joueur posÃ¨de il la couleur demandÃ©e ?

< l'entame est en atout

< Le joueur a il un ajout de valeur supÃ©rieur au meilleur ajout posÃ©

< si la carte est un ajout le valeur supÃ©rieur au meilleur atout posÃ©

< Si la carte est un atout

< Si la carte est dans la couleur demandÃ©e

< Le partemenaire est maitre ?

< Le joueur a il un atout ?

< il y a deja un ajout de posÃ©

< Le joueur a il un ajout de valeur supÃ©rieur au meilleur ajout posÃ©

< si la carte est un ajout le valeur supÃ©rieur au meilleur atout posÃ©

< Si la carte est un atout alors elle est valide

Définition à la ligne 261 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.11.2.3 distribueCarte()

```
void distribueCarte (
            Carte * pCarteMain )
```

distribue les cartes entre les différent joueur

**Paramètres**

| | |
|---|---|
| *Carte* | *pCarteMain : pointeur sur le tableau qui stocke les carte dans la main de chaque joueur |
| | |

Définition à la ligne 12 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.11.2.4 forceCarte()

```
float forceCarte (
            Carte carteACalculer,
            Couleur atout,
            Couleur entame )
```

calcul la force d'une carte, en faisant la probabilitée qu'elle a de gagné contre toutes les autres cartes

**Paramètres**

| | |
|---|---|
| *Carte* | carteACalculer : carte dont on veut connaitre la force |
| *Couleur* | atout : couleur de l'atout durant la partie |
| *Couleur* | entame : couleur de l'entame durant la partie |

**Renvoie**

float : probabilitée de victoir de la carte

< la carte est un atout ou que la manche se joue en tout atout

< si on est en tout atout mais que la couleur de la carte n'est pa la couleur de l'entame alors la carte sera plus faible que toutes les cartees dans la bonne couleur

< la manche est en sans atout et la carte est dans la couleur de l'entame

< on est en atout d'une certaine couleur ou en sans atoutavec une couleur diffÃ©rente de l'entame

< quand l'entame est diffÃ©rent de la couleur de la carte et que ce n'est pas un atout alor on pert a chaque fois

Définition à la ligne 80 du fichier gestionCarte.c.

Voici le graphe des appelants de cette fonction :

#### 4.11.2.5 rechercheAnnonce()

```
void rechercheAnnonce (
            Carte * pCarteMain,
            int pointAnonce[],
            Joueur utilisateur,
            char * pseudo[] )
```

rechere dans un tableau de cartes qui correspond a la main d'un joureur si il peut faire des anonc, et si oui ajoute les points assacier au tableau dans le tableau des anonce

**Paramètres**

| | |
|---|---|
| *Carte* | ∗pCarteMain : pointeur sur le premier élément du tableau qui cintient les cartes de tout les joueur |
| *int* | pointAnonce[] : tableau contenant les point d'anonce |
| *Joueur* | utilisateur : donne la position de l'utilisateur, mettre SANS_Joueur pour faire une partie avec uniquement des ordinateur |
| *char* | ∗pseudo[] : tableau contenant les pseudo des 4 joueur |

**Renvoie**

void

< on cherche si le joueur a un carrÃ©

Définition à la ligne 435 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.11.2.6 rechercherCarte()

```
char rechercherCarte (
            Carte * pCarte,
            int nbCarte,
            Couleur couleurCherche,
            Valeur valeurCherche )
```

cherche dans un tableau de carte si il y a une certaine couleur ou une certaine valeur ou les deux

**Paramètres**

| | |
|---|---|
| *Carte* | ∗pCarte : pointeur sur un tableau de carte |
| *int* | nbCarte : nombre de carte dans le tableau |
| *Couleur* | couleurCherche : couleur que l'on veut chercher dans le tableau (mettre SANS_COULEUR pour rechercher toutes les couleur) |
| *Valeur* | valeurCherche : valeur que l'on cherche dans le tableau (mettre SANS_VALEUR pour chercher toutes les valeur |

**Renvoie**

char 1 si on a trouvé une carte qui correspond au critère 0 sinon

Définition à la ligne 343 du fichier gestionCarte.c.

Voici le graphe des appelants de cette fonction :

#### 4.11.2.7 rechercherCarteSuperieur()

```
char rechercherCarteSuperieur (
            Carte * pCarte,
            int nbCarte,
```

```
            Carte carteCherche,
            Couleur atout,
            Couleur entame )
```

cherche dans un tableau de carte si il y a une carte d'une force suppérieur a carteCherche

**Paramètres**

| | |
|---|---|
| *Carte* | ∗pCarte : pointeur sur un tableau de carte |
| *int* | nbCarte : nombre de carte dans le tableau |
| *Carte* | carteCherche : carte a laquele on veut que la carte cherchée soit supérieur |
| *Couleur* | atout : couleur de l'atout dans la manche |
| *Couleur* | entame : couleur de l'entame dans la manche |

**Renvoie**

char 1 si on a trouvé une carte qui correspond au critère 0 sinon

Définition à la ligne 355 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.11.2.8 setCarte()

```
void setCarte (
            Carte * carte,
            Valeur valeurCarte,
            Couleur couleurCarte )
```

donne une valeur et une couleur a une carte

**Paramètres**

| | |
|---|---|
| *Carte* | ∗carte : pointeur sur la carte a modifier |
| *Valeur* | valeurCarte : valeur a donner a la carte |
| *Couleur* | couleurCarte : couleur a donner a la carte |

**Renvoie**

void

Définition à la ligne 50 du fichier gestionCarte.c.

Voici le graphe des appelants de cette fonction :

#### 4.11.2.9 sommeForceCarte()

```
float sommeForceCarte (
            Carte * tableauCarte,
            int nbCarte,
            Couleur atout )
```

calucule la somme force des carte dans un tableau

**Paramètres**

| | |
|---|---|
| *Carte* | tableauCarte : tableau de carte sur lequel on veut travailler |
| *int* | nbCarte : nombre de carte dans le tableau |
| *Couleur* | atout : couleur de l'atout sur la manche |

**Renvoie**

float : la valeur de lla somme des force des carte

Définition à la ligne 369 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.11.2.10 supprimeCarte()

```
void supprimeCarte (
            Carte carte[],
            int nbCarte,
            int carteASupprimer )
```

supprime une carte d'un tableau et réduit la taille de ce tableu par le bas

**Paramètres**

| | |
|---|---|
| *Carte* | carte[] : tableau conteneant les carte a modifier |
| *int* | nbCarte : nombre total de carte |
| *int* | carteASupprimer : indice dans le tableau de la carte a enlever |

**Renvoie**

void

Définition à la ligne 56 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.11.2.11 trieCarte()

```
void trieCarte (
            Carte tableauCarte[],
            int nbCarte,
            Couleur atout )
```

trie un tableau de cartes

**Paramètres**

| | |
|---|---|
| *Carte* | tableauCarte[] : tableau contenant les cartes a trier |
| *int* | nbCarte nombre de carte dans le tableau |
| *Couleur* | atout : couleur de l'atout dans la manche |

**Renvoie**

void

< On rÃ©alise un tri a bulle pour mettre atout du plus fort au moin fort au dÃ©but de la main puis les carte par famille

< Si la carte dois aller devant dans le tableau

Définition à la ligne 386 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.11.2.12 vainqueurPli()

```
Joueur vainqueurPli (
            Carte pli[],
            Couleur atout,
            Joueur premierAJouer )
```

donne le vainquer d'un pli

**Paramètres**

| | |
|---|---|
| *Carte* | pli[] : tableau contenant les carte du pli |
| *Couleur* | atout : couleur de l'atout durant cette manche |
| *Joueur* | premierAJouer : joueur qui pose la première carte du pli |

**Renvoie**

Joueur : le jouer qui a gagné le pli

< La carte du joueur que l'on test est plus forte que la meilleur actuelle alors ca devient la meilleure

Définition à la ligne 64 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

## 4.12 Référence du fichier gestionCarte.h

ficher header contenant les protypes des fonctions relative a la gestion des cartes

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

### Fonctions

- void distribueCarte (Carte *pCarteMain)
  *distribue les cartes entre les différent joueur*
- void setCarte (Carte *carte, Valeur valeurCarte, Couleur couleurCarte)
  *donne une valeur et une couleur a une carte*
- void supprimeCarte (Carte carte[ ], int nbCarte, int carteASupprimer)
  *supprime une carte d'un tableau et réduit la taille de ce tableu par le bas*
- Joueur vainqueurPli (Carte pli[ ], Couleur atout, Joueur premierAJouer)
  *donne le vainquer d'un pli*
- float forceCarte (Carte carteACalculer, Couleur atout, Couleur entame)
  *calcul la force d'une carte, en faisant la probabilitée qu'elle a de gagné contre toutes les autres cartes*
- char carteValide (Carte cartePose, Carte pli[ ], Couleur atout, Carte *pCarteMainJoueur, Joueur premierA↩ Jouer, Joueur parle)
  *ddétermine si une carte peut etre posé par un joueur*
- char rechercherCarte (Carte *pCarte, int nbCarte, Couleur couleurCherche, Valeur valeurCherche)

*cherche dans un tableau de carte si il y a une certaine couleur ou une certaine valeur ou les deux*

— char rechercherCarteSuperieur (Carte ∗pCarte, int nbCarte, Carte carteCherche, Couleur atout, Couleur entame)

*cherche dans un tableau de carte si il y a une carte d'une force suppérieur a carteCherche*

— float sommeForceCarte (Carte ∗tableauCarte, int nbCarte, Couleur atout)

*calucule la somme force des carte dans un tableau*

— void trieCarte (Carte tableauCarte[ ], int nbCarte, Couleur atout)

*trie un tableau de cartes*

— char cartePlaceAvant (Carte carteRefference, Carte carteCompare, Couleur atout)

*permet de savoir si une carte doit etre placée avant lors du trie des cartes*

— void rechercheAnnonce (Carte ∗pCarteMain, int pointAnonce[ ], Joueur utilisateur, char ∗pseudo[ ])

*rechere dans un tableau de cartes qui correspond a la main d'un joureur si il peut faire des anonc, et si oui ajoute les points assacier au tableau dans le tableau des anonce*

### 4.12.1 Description détaillée

ficher header contenant les protypes des fonctions relative a la gestion des cartes

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.12.2 Documentation des fonctions

#### 4.12.2.1 cartePlaceAvant()

```
char cartePlaceAvant (
            Carte carteRefference,
            Carte carteCompare,
            Couleur atout )
```

permet de savoir si une carte doit etre placée avant lors du trie des cartes

**Paramètres**

| | |
|---|---|
| *Carte* | carteRefference : carte par raport a laquelle on compare |
| *Carte* | carteCompare : carte que l'on compare |
| *Couleur* | atout : couleur de l'atout dans la manche |

**Renvoie**

char : renvoie 1 si la catre doit etre placée avant 0 sinon

< Les deux carte sont elle de la meme famille

< si carteCompate est un atout

< si la couleur de la carteCompare est devant celle de carteRefference dans l'Ã©numÃ©ration(ordre arbitraire)

Définition à la ligne 404 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.12.2.2 carteValide()

```
char carteValide (
            Carte cartePose,
            Carte pli[],
            Couleur atout,
            Carte * pCarteMainJoueur,
            Joueur premierAJouer,
            Joueur parle )
```

ddétermine si une carte peut etre posé par un joueur

**Paramètres**

| | |
|---|---|
| *Carte* | cartPose : carte que l'on veut poser |
| *Carte* | pli[] : tableau contenet les carte du pli |
| *Couleur* | atout : couleur de l'atout |
| *Carte* | ∗pCarteMainJoueur pointeur sur un tableau de 8 carte contenent les cartes dans la main du joueur |
| *Joueur* | premierAJouer : premier joueur a jouer dans le pli |
| *Joueur* | parle : joueur en train de jouer |

**Renvoie**

char 1 si la carte est valide 0 sinon

< Fonction faite a partir de l'oranigrame qui montre comment dÃ©terminer si une carte est valide a partir des rÃ¨gle

< On verifier que ce n'est pas une carte vide

< premiÃ¨re carte du plis ?

< Le joueur posÃ¨de il la couleur demandÃ©e ?

< l'entame est en atout

< Le joueur a il un ajout de valeur supÃ©rieur au meilleur ajout posÃ©

< si la carte est un ajout le valeur supÃ©rieur au meilleur atout posÃ©

< Si la carte est un atout

< Si la carte est dans la couleur demandÃ©e

< Le partemenaire est maitre ?

< Le joueur a il un atout ?

< il y a deja un ajout de posÃ©

< Le joueur a il un ajout de valeur supÃ©rieur au meilleur ajout posÃ©

< si la carte est un ajout le valeur supÃ©rieur au meilleur atout posÃ©

< Si la carte est un atout alors elle est valide

Définition à la ligne 261 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.12.2.3 distribueCarte()

```
void distribueCarte (
            Carte * pCarteMain )
```

distribue les cartes entre les différent joueur

**Paramètres**

| | |
|---|---|
| *Carte* | *pCarteMain : pointeur sur le tableau qui stocke les carte dans la main de chaque joueur |
| | |

Définition à la ligne 12 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.12.2.4 forceCarte()

```
float forceCarte (
            Carte carteACalculer,
            Couleur atout,
            Couleur entame )
```

calcul la force d'une carte, en faisant la probabilitée qu'elle a de gagné contre toutes les autres cartes

**Paramètres**

| | |
|---|---|
| *Carte* | carteACalculer : carte dont on veut connaitre la force |
| *Couleur* | atout : couleur de l'atout durant la partie |
| *Couleur* | entame : couleur de l'entame durant la partie |

**Renvoie**

float : probabilitée de victoir de la carte

< la carte est un atout ou que la manche se joue en tout atout

< si on est en tout atout mais que la couleur de la carte n'est pa la couleur de l'entame alors la carte sera plus faible que toutes les cartees dans la bonne couleur

< la manche est en sans atout et la carte est dans la couleur de l'entame

< on est en atout d'une certaine couleur ou en sans atoutavec une couleur diffÃ©rente de l'entame

< quand l'entame est diffÃ©rent de la couleur de la carte et que ce n'est pas un atout alor on pert a chaque fois

Définition à la ligne 80 du fichier gestionCarte.c.

Voici le graphe des appelants de cette fonction :

#### 4.12.2.5 rechercheAnnonce()

```
void rechercheAnnonce (
            Carte * pCarteMain,
            int pointAnonce[],
            Joueur utilisateur,
            char * pseudo[] )
```

rechere dans un tableau de cartes qui correspond a la main d'un joureur si il peut faire des anonc, et si oui ajoute les points assacier au tableau dans le tableau des anonce

**Paramètres**

| | |
|---|---|
| *Carte* | ∗pCarteMain : pointeur sur le premier élément du tableau qui cintient les cartes de tout les joueur |
| *int* | pointAnonce[] : tableau contenant les point d'anonce |
| *Joueur* | utilisateur : donne la position de l'utilisateur, mettre SANS_Joueur pour faire une partie avec uniquement des ordinateur |
| *char* | ∗pseudo[] : tableau contenant les pseudo des 4 joueur |

**Renvoie**

void

< on cherche si le joueur a un carrÃ©

Définition à la ligne 435 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.12.2.6 rechercherCarte()

```
char rechercherCarte (
            Carte * pCarte,
            int nbCarte,
            Couleur couleurCherche,
            Valeur valeurCherche )
```

cherche dans un tableau de carte si il y a une certaine couleur ou une certaine valeur ou les deux

**Paramètres**

| | |
|---|---|
| *Carte* | ∗pCarte : pointeur sur un tableau de carte |
| *int* | nbCarte : nombre de carte dans le tableau |
| *Couleur* | couleurCherche : couleur que l'on veut chercher dans le tableau (mettre SANS_COULEUR pour rechercher toutes les couleur) |
| *Valeur* | valeurCherche : valeur que l'on cherche dans le tableau (mettre SANS_VALEUR pour chercher toutes les valeur |

**Renvoie**

char 1 si on a trouvé une carte qui correspond au critère 0 sinon

Définition à la ligne 343 du fichier gestionCarte.c.

Voici le graphe des appelants de cette fonction :

#### 4.12.2.7 rechercherCarteSuperieur()

```
char rechercherCarteSuperieur (
            Carte * pCarte,
            int nbCarte,
            Carte carteCherche,
            Couleur atout,
            Couleur entame )
```

cherche dans un tableau de carte si il y a une carte d'une force suppérieur a carteCherche

**Paramètres**

| | |
|---|---|
| *Carte* | ∗pCarte : pointeur sur un tableau de carte |
| *int* | nbCarte : nombre de carte dans le tableau |
| *Carte* | carteCherche : carte a laquele on veut que la carte cherchée soit supérieur |
| *Couleur* | atout : couleur de l'atout dans la manche |
| *Couleur* | entame : couleur de l'entame dans la manche |

**Renvoie**

char 1 si on a trouvé une carte qui correspond au critère 0 sinon

Définition à la ligne 355 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.12.2.8 setCarte()

```
void setCarte (
            Carte * carte,
            Valeur valeurCarte,
            Couleur couleurCarte )
```

donne une valeur et une couleur a une carte

**Paramètres**

| | |
|---|---|
| *Carte* | ∗carte : pointeur sur la carte a modifier |
| *Valeur* | valeurCarte : valeur a donner a la carte |
| *Couleur* | couleurCarte : couleur a donner a la carte |

**Renvoie**

void

Définition à la ligne 50 du fichier gestionCarte.c.

Voici le graphe des appelants de cette fonction :

#### 4.12.2.9 sommeForceCarte()

```
float sommeForceCarte (
            Carte * tableauCarte,
            int nbCarte,
            Couleur atout )
```

calucule la somme force des carte dans un tableau

**Paramètres**

| | |
|---|---|
| *Carte* | tableauCarte : tableau de carte sur lequel on veut travailler |
| *int* | nbCarte : nombre de carte dans le tableau |
| *Couleur* | atout : couleur de l'atout sur la manche |

**Renvoie**

float : la valeur de lla somme des force des carte

Définition à la ligne 369 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.12.2.10 supprimeCarte()

```
void supprimeCarte (
            Carte carte[],
            int nbCarte,
            int carteASupprimer )
```

supprime une carte d'un tableau et réduit la taille de ce tableu par le bas

**Paramètres**

| | |
|---|---|
| *Carte* | carte[] : tableau conteneant les carte a modifier |
| *int* | nbCarte : nombre total de carte |
| *int* | carteASupprimer : indice dans le tableau de la carte a enlever |

**Renvoie**

void

Définition à la ligne 56 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.12.2.11 trieCarte()

```
void trieCarte (
            Carte tableauCarte[],
            int nbCarte,
            Couleur atout )
```

trie un tableau de cartes

**Paramètres**

| | |
|---|---|
| *Carte* | tableauCarte[] : tableau contenant les cartes a trier |
| *int* | nbCarte nombre de carte dans le tableau |
| *Couleur* | atout : couleur de l'atout dans la manche |

**Renvoie**

void

< On rÃ©alise un tri a bulle pour mettre atout du plus fort au moin fort au dÃ©but de la main puis les carte par famille

< Si la carte dois aller devant dans le tableau

Définition à la ligne 386 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.12.2.12 vainqueurPli()

```
Joueur vainqueurPli (
            Carte pli[],
            Couleur atout,
            Joueur premierAJouer )
```

donne le vainquer d'un pli

**Paramètres**

| | |
|---|---|
| *Carte* | pli[] : tableau contenant les carte du pli |
| *Couleur* | atout : couleur de l'atout durant cette manche |
| *Joueur* | premierAJouer : joueur qui pose la première carte du pli |

**Renvoie**

Joueur : le jouer qui a gagné le pli

< La carte du joueur que l'on test est plus forte que la meilleur actuelle alors ca devient la meilleure

Définition à la ligne 64 du fichier gestionCarte.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

## 4.13 Référence du fichier gestionFichier.c

ficher source contenant les fonctions relative à la gestion des fichiers

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

Graphe des dépendances par inclusion de gestionFichier.c:

### Fonctions

- int ecritPseudo (char *pPseudo, FILE *pFichier)
  *fonction qui recherche dans un fichier si le joueur est deja enregistrÃ© au le rajoute Ã  la fin si ce n'est pas le cas*
- int ecrireStatistique (FILE *fichier, int ligne, int statAModifier, int type)
  *modifie la valeur d'une statistique dans le fichier de sauvegarde des scores*
- int ecrireLeaderboard (FILE *fichier, char *pseudo, int scoreJ)
  *vÃ©rifie si un nouveau record Ã  Ã©tÃ© battu et l'ajoute si c'est le cas*

### 4.13.1 Description détaillée

ficher source contenant les fonctions relative à la gestion des fichiers

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.13.2 Documentation des fonctions

#### 4.13.2.1 ecrireLeaderboard()

```
int ecrireLeaderboard (
            FILE * fichier,
            char * pseudo,
            int score )
```

vÃ©rifie si un nouveau record Ã  Ã©tÃ© battu et l'ajoute si c'est le cas

**Paramètres**

| | |
|---|---|
| *FILE* | ∗fichier : pointeur vers fichier de meilleurs score |
| *char* | ∗pPseudo : pointeur vers le pseudo du joueur |
| *int* | score : score a Ã©ventuellement enregistrer |

**Renvoie**

0

Définition à la ligne 97 du fichier gestionFichier.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.13.2.2 ecrireStatistique()

```
int ecrireStatistique (
            FILE * fichier,
            int ligne,
            int statAModifier,
            int type )
```

modifie la valeur d'une statistique dans le fichier de sauvegarde des scores

**Paramètres**

| | |
|---|---|
| *FILE* | ∗fichier : pointeur vers le fichier |
| *int* | ligne : ligne du fichier Ã  laquelle sont sauvegardÃ©e les statistiques du joueur |
| *int* | statAModifier : la nouvelle statistique Ã  enregistrer |
| *int* | type : vaut 1 pour mdifier le nombre de victoires, vaut 2 pour changer le score max, vaut 3 pour changer le nombre de manche min pour une victoire |

**Renvoie**

valeur enregistrÃ©e

Définition à la ligne 44 du fichier gestionFichier.c.

Voici le graphe des appelants de cette fonction :

#### 4.13.2.3 ecriturePseudo()

```
int ecriturePseudo (
            char * pPseudo,
            FILE * pFichier )
```

fonction qui recherche dans un fichier si le joueur est deja enregistrÃ© au le rajoute Ã  la fin si ce n'est pas le cas

**Paramètres**

| | |
|---|---|
| *char* | ∗pPseudo: pointeur vers le nom du joueur |
| *FILE* | ∗pFichier : pointeur vers le fichier contenant les statistiques |

**Renvoie**

le numero de la ligne dans lequel est notÃ© le pseudo

Définition à la ligne 11 du fichier gestionFichier.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

## 4.14 Référence du fichier gestionFichier.h

ficher header contenant les prototypes des fonctions relative Ã  la gestion des fichiers

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

### Fonctions

- int ecritirePseudo (char ∗pPseudo, FILE ∗pFichier)
  *fonction qui recherche dans un fichier si le joueur est deja enregistrÃ© au le rajoute Ã  la fin si ce n'est pas le cas*
- int ecrireStatistique (FILE ∗fichier, int ligne, int statAModifier, int type)
  *modifie la valeur d'une statistique dans le fichier de sauvegarde des scores*
- int ecrireLeaderboard (FILE ∗fichier, char ∗pseudo, int score)
  *vÃ©rifie si un nouveau record Ã  Ã©tÃ© battu et l'ajoute si c'est le cas*

### 4.14.1 Description détaillée

ficher header contenant les prototypes des fonctions relative Ã  la gestion des fichiers

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.14.2 Documentation des fonctions

#### 4.14.2.1 ecrireLeaderboard()

```
int ecrireLeaderboard (
            FILE * fichier,
            char * pseudo,
            int scoreJ )
```

vÃ©rifie si un nouveau record Ã  Ã©tÃ© battu et l'ajoute si c'est le cas

**Paramètres**

| | |
|---|---|
| *FILE* | ∗fichier : pointeur vers fichier de meilleurs score |
| *char* | ∗pPseudo : pointeur vers le pseudo du joueur |
| *int* | score : score a Ã©ventuellement enregistrer |

**Renvoie**

0

Définition à la ligne 97 du fichier gestionFichier.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.14.2.2 ecrireStatistique()

```
int ecrireStatistique (
            FILE * fichier,
            int ligne,
            int statAModifier,
            int type )
```

modifie la valeur d'une statistique dans le fichier de sauvegarde des scores

**Paramètres**

| | |
|---|---|
| *FILE* | ∗fichier : pointeur vers le fichier |
| *int* | ligne : ligne du fichier Ã  laquelle sont sauvegardÃ©e les statistiques du joueur |
| *int* | statAModifier : la nouvelle statistique Ã  enregistrer |
| *int* | type : vaut 1 pour mdifier le nombre de victoires, vaut 2 pour changer le score max, vaut 3 pour changer le nombre de manche min pour une victoire |

**Renvoie**

valeur enregistrÃ©e

Définition à la ligne 44 du fichier gestionFichier.c.

Voici le graphe des appelants de cette fonction :

#### 4.14.2.3 ecriturePseudo()

```
int ecriturePseudo (
            char * pPseudo,
            FILE * pFichier )
```

fonction qui recherche dans un fichier si le joueur est deja enregistrÃ© au le rajoute Ã  la fin si ce n'est pas le cas

**Paramètres**

| | |
|---|---|
| *char* | ∗pPseudo: pointeur vers le nom du joueur |
| *FILE* | ∗pFichier : pointeur vers le fichier contenant les statistiques |

**Renvoie**

le numero de la ligne dans lequel est notÃ© le pseudo

Définition à la ligne 11 du fichier gestionFichier.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

## 4.15 Référence du fichier ia.c

ficher source contenant les fonctions relative au inteligences artificielles

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

Graphe des dépendances par inclusion de ia.c:

### Fonctions

- Contrat proposeContratIa (Joueur parle, Carte ∗pCarteMain, Contrat dernierContrat)
  *proposition par l'ia d'un contrat*
- int choixCarteIA (Joueur joueur, Carte ∗pMainJoueur, Carte pli[ ], Joueur premierJoueur, Couleur atout, int carteRestante)
  *proposition de la pose d'une carte par l'IA*

### 4.15.1 Description détaillée

ficher source contenant les fonctions relative au inteligences artificielles

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.15.2 Documentation des fonctions

#### 4.15.2.1 choixCarteIA()

```
int choixCarteIA (
            Joueur joueur,
            Carte * pMainJoueur,
            Carte pli[],
            Joueur dernierVainqueur,
            Couleur atout,
            int carteRestante )
```

proposition de la pose d'une carte par l'IA

**Paramètres**

| | |
|---|---|
| *Joueur* | joueur : determine qu'elle IA doit jouer |
| *Carte* | ∗pMainJoueur : pointeur vers le tableau qui enregistre la main de l'IA |
| *Carte* | pli[] : tableau qui enregistre les cartes jouées pendant le pli |
| *Joueur* | dernierVainqueur : defini quel joueur à posé la première carte |
| *Couleur* | atout : defini quelle couleur est en atout |
| *int* | carteRestante : nombre de cartes resantes en main |

**Renvoie**

debug for(int i=0;i<nbCarteValide;i++){ printf("|%d| \n",numCarteValide[i]); } afficheMain(pCarteValidee);

debug for(int i=0;i<offset;i++){ printf("|%d|| \n",numCarteGagnante[i]); } afficheMain(pCarteGagante);

Définition à la ligne 60 du fichier ia.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.15.2.2 proposeContratIa()

```
Contrat proposeContratIa (
            Joueur parle,
            Carte * pCarteMain,
            Contrat dernierContrat )
```

proposition par l'ia d'un contrat

**Paramètres**

| | |
|---|---|
| *Joueur* | parle : joueur qui parle |
| *Carte* | ∗pCarteMain pointeur sur le tableau qui contient les cartes du joueur |
| *Contrat* | dernierContrat : dernier contrat qui a été proposé |

**Renvoie**

Contrat

< Permet de dÃ©terminer de maniÃ¨re euristique la meilleur valeur de seuilMinPrise et seuilMaxPrise

< ON coinche !

Définition à la ligne 11 du fichier ia.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

## 4.16 Référence du fichier ia.h

ficher header contenant les prototypes des fonctions relative au inteligences artificielles

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

## Fonctions

- Contrat proposeContratIa (Joueur parle, Carte ∗pCarteMain, Contrat dernierContrat)
  *proposition par l'ia d'un contrat*
- int choixCarteIA (Joueur joueur, Carte ∗pMainJoueur, Carte pli[ ], Joueur dernierVainqueur, Couleur atout, int carteRestante)
  *proposition de la pose d'une carte par l'IA*

### 4.16.1 Description détaillée

ficher header contenant les prototypes des fonctions relative au inteligences artificielles

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.16.2 Documentation des fonctions

#### 4.16.2.1 choixCarteIA()

```
int choixCarteIA (
            Joueur joueur,
            Carte * pMainJoueur,
            Carte pli[],
            Joueur premierJoueur,
            Couleur atout,
            int carteRestante )
```

proposition de la pose d'une carte par l'IA

**Paramètres**

| | |
|---|---|
| *Joueur* | joueur : determine qu'elle IA doit jouer |
| *Carte* | ∗pMainJoueur : pointeur vers le tableau qui enregistre la main de l'IA |
| *Carte* | pli[] : tableau qui enregistre les cartes jouées pendant le pli |
| *Joueur* | dernierVainqueur : defini quel joueur à posé la première carte |
| *Couleur* | atout : defini quelle couleur est en atout |
| *int* | carteRestante : nombre de cartes resantes en main |

**Renvoie**

debug for(int i=0;i<nbCarteValide;i++){ printf("|%d| \n",numCarteValide[i]); } afficheMain(pCarteValidee);

debug for(int i=0;i<offset;i++){ printf("|%d|| \n",numCarteGagnante[i]); } afficheMain(pCarteGagante);

Définition à la ligne 60 du fichier ia.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.16.2.2 proposeContratIa()

```
Contrat proposeContratIa (
            Joueur parle,
            Carte * pCarteMain,
            Contrat dernierContrat )
```

proposition par l'ia d'un contrat

**Paramètres**

| | |
|---|---|
| *Joueur* | parle : joueur qui parle |
| *Carte* | ∗pCarteMain pointeur sur le tableau qui contient les cartes du joueur |
| *Contrat* | dernierContrat : dernier contrat qui a été proposé |

**Renvoie**

Contrat

< Permet de dÃ©terminer de maniÃ¨re euristique la meilleur valeur de seuilMinPrise et seuilMaxPrise

< ON coinche !

Définition à la ligne 11 du fichier ia.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

## 4.17 Référence du fichier main.c

ficher contenant la fonction main

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

Graphe des dépendances par inclusion de main.c:

### Fonctions

— int main (int argc, char ∗argv[ ])
*Entrée du programme.*

### 4.17.1 Description détaillée

ficher contenant la fonction main

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.17.2 Documentation des fonctions

#### 4.17.2.1 main()

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

Entrée du programme.

**Auteur**

Florian.C

**Renvoie**

EXIT_SUCCESS - Arrêt normal du programme

< CODE FINAL

< definit la taille de la fennetre a 50 lignes et 91 colones

< FIN CODE FINAL

Définition à la ligne 19 du fichier main.c.

Voici le graphe d'appel pour cette fonction :

## 4.18 Référence du fichier main.h

ficher la dÃ©claration des contrante, les Ã©numÃ©rations, les structe et l'inclusion des haeder du projet

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <time.h>
#include <string.h>
#include "tableau.h"
#include "acquisition.h"
#include "affichage.h"
#include "gestionCarte.h"
#include "general.h"
#include "formatageChaine.h"
#include "autre.h"
#include "ia.h"
#include "sous-menus.h"
#include "gestionFichier.h"
```

Graphe des dépendances par inclusion de main.h: Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

### Structures de données

- struct Carte
  *represente une carte d'un jeu de 32 cartes*
- struct Contrat
  *represente un contrat Ã  la belote coichÃ©e*

### Macros

- #define TAILLE_MAXI_PESEUDO 20
- #define TAILLE_MAXI_COULEUR 8
- #define NB_JOUEUR 4
- #define TAILLE_MAXI_MESSAGE 500
- #define NB_TOATAL_CARTE 32
- #define NIVEAU_IA 2
- #define NB_CARRACTERE_SCORE 33
- #define POSITION_NB_VICTOIRE 21
- #define POSITION_SCORE_MAX 25
- #define POSITION_NB_MANCHES_POUR_GAGNER 30
- #define NB_CARRACTERE_LEADERBOARD 26
- #define POSITION_RECORD_VICTOIRE 21
- #define DEBUG_MODE 0
- #define MODE_1_MANCHE 0

### Définitions de type

- typedef enum Couleur Couleur
- typedef enum Valeur Valeur
- typedef enum Joueur Joueur
- typedef enum Coinche Coinche
- typedef enum NbPoint NbPoint
- typedef struct Carte Carte
- typedef struct Contrat Contrat
- typedef enum TypeMessage TypeMessage

## Énumérations

- enum Couleur {
  SANS_COULEUR = 0, COEUR = 1, PIQUE = 2, CARREAU = 3,
  TREFLE = 4, TOUT_ATOUT = 5, SANS_ATOUT = 6 }
  *Les diffÃ©rentes couleur d'atout possible a la belote coinchÃ©e.*
- enum Valeur {
  SANS_VALEUR = 0, AS = 1, VALET = 11, DAME = 12,
  ROI = 13, SEPT = 7, HUIT = 8, NEUF = 9,
  DIX = 10 }
  *Les diffÃ©rentes valeur possible des cartes d'un jeu de 32 cartes.*
- enum Joueur {
  SANS_JOUEUR = 0, NORD = 1, EST = 2, SUD = 3,
  OUEST = 4 }
  *Les diffÃ©rente joueur lors d'une partie.*
- enum Coinche { NORMAL = 0, COINCHE = 1, SURCOINCHE = 2 }
  *indique il un contrat est coinchÃ© ou surcoinchÃ©*
- enum NbPoint {
  ZERO = 0, QUATRE_VINGT = 80, QUATRE_VINGT_DIX = 90, CENT = 100,
  CENT_DIX = 110, CENT_VINGT = 120, CENT_TRENTE = 130, CENT_QUARANTE = 140,
  CENT_CINQUANTE = 150, CENT_SOIXANTE = 160, CAPOT = 170, GENERALE = 180 }
- enum TypeMessage { SANS_MESSAGE, POSE_CARTE, RESULTAT_PLI, RESULTAT_MANCHE }
  *contante Ã  passer en paramÃ¨tre Ã  la fonction gÃ©nÃ¨reMessage() pour donner le type de message a gÃ©nÃ©rer*

### 4.18.1 Description détaillée

ficher la dÃ©claration des contrante, les Ã©numÃ©rations, les structe et l'inclusion des haeder du projet

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.18.2 Documentation des macros

#### 4.18.2.1 DEBUG_MODE

```
#define DEBUG_MODE 0
```

0 pour le mode normal, 1 pour le mode dÃ©bug

Définition à la ligne 26 du fichier main.h.

#### 4.18.2.2 MODE_1_MANCHE

```
#define MODE_1_MANCHE 0
```

0 pour le mode normal, 1 pour le mode ou une partie est constituÃ© d'une uique manche Structure et Ã©numÃ©rations

Définition à la ligne 29 du fichier main.h.

#### 4.18.2.3 NB_CARRACTERE_LEADERBOARD

```
#define NB_CARRACTERE_LEADERBOARD 26
```

Définition à la ligne 24 du fichier main.h.

#### 4.18.2.4 NB_CARRACTERE_SCORE

```
#define NB_CARRACTERE_SCORE 33
```

Définition à la ligne 20 du fichier main.h.

#### 4.18.2.5 NB_JOUEUR

```
#define NB_JOUEUR 4
```

Définition à la ligne 16 du fichier main.h.

#### 4.18.2.6 NB_TOATAL_CARTE

```
#define NB_TOATAL_CARTE 32
```

Définition à la ligne 18 du fichier main.h.

#### 4.18.2.7 NIVEAU_IA

```
#define NIVEAU_IA 2
```

Définition à la ligne 19 du fichier main.h.

#### 4.18.2.8 POSITION_NB_MANCHES_POUR_GAGNER

```
#define POSITION_NB_MANCHES_POUR_GAGNER 30
```

Définition à la ligne 23 du fichier main.h.

#### 4.18.2.9 POSITION_NB_VICTOIRE

```
#define POSITION_NB_VICTOIRE 21
```

Définition à la ligne 21 du fichier main.h.

#### 4.18.2.10 POSITION_RECORD_VICTOIRE

```
#define POSITION_RECORD_VICTOIRE 21
```

Définition à la ligne 25 du fichier main.h.

#### 4.18.2.11 POSITION_SCORE_MAX

```
#define POSITION_SCORE_MAX 25
```

Définition à la ligne 22 du fichier main.h.

#### 4.18.2.12 TAILLE_MAXI_COULEUR

```
#define TAILLE_MAXI_COULEUR 8
```

Définition à la ligne 15 du fichier main.h.

#### 4.18.2.13 TAILLE_MAXI_MESSAGE

```
#define TAILLE_MAXI_MESSAGE 500
```

Définition à la ligne 17 du fichier main.h.

#### 4.18.2.14 TAILLE_MAXI_PESEUDO

```
#define TAILLE_MAXI_PESEUDO 20
```

< Constantes

Définition à la ligne 14 du fichier main.h.

### 4.18.3 Documentation des définitions de type

#### 4.18.3.1 Carte

```
typedef struct Carte Carte
```

#### 4.18.3.2 Coinche

```
typedef enum Coinche Coinche
```

#### 4.18.3.3 Contrat

```
typedef struct Contrat Contrat
```

#### 4.18.3.4 Couleur

```
typedef enum Couleur Couleur
```

#### 4.18.3.5 Joueur

```
typedef enum Joueur Joueur
```

#### 4.18.3.6 NbPoint

```
typedef enum NbPoint NbPoint
```

#### 4.18.3.7 TypeMessage

```
typedef enum TypeMessage TypeMessage
```

Fontions standard Fontions crÃ©Ã©e dans le cardre du projet

#### 4.18.3.8 Valeur

```
typedef enum Valeur Valeur
```

### 4.18.4 Documentation du type de l'énumération

#### 4.18.4.1 Coinche

```
enum Coinche
```

indique il un contrat est coinchÃ© ou surcoinchÃ©

**Valeurs énumérées**

| | |
|---|---|
| NORMAL | |
| COINCHE | Le contrat n'est pas coichÃ©, par dÃ©faut |
| SURCOINCHE | Le contrat est CoinchÃ© |

Définition à la ligne 97 du fichier main.h.

#### 4.18.4.2 Couleur

```
enum Couleur
```

Les diffÃ©rentes couleur d'atout possible a la belote coinchÃ©e.

toute les couleur disponible ainsi que tout atouts et sans atout sert a la fois pour le type Carte et le type Contrat

**Valeurs énumérées**

| | |
|---|---|
| SANS_COULEUR | |
| COEUR | Couleur par dÃ©faut, pour une carte vide |
| PIQUE | Couleur coeur |
| CARREAU | Couleur pique |
| TREFLE | Couleur carreau |
| TOUT_ATOUT | Couleur trefle |
| SANS_ATOUT | Tout atout |

Définition à la ligne 39 du fichier main.h.

#### 4.18.4.3 Joueur

enum Joueur

Les diffÃ©rente joueur lors d'une partie.

Les joueur sont nomÃ© par les different point cardinaut correspondant Ã  leur placement sur la table

**Valeurs énumérées**

| | |
|---|---|
| SANS_JOUEUR | |
| NORD | Joueur par dÃ©faut, quand il n'y a pas de joueur ou pour l'initialitation |
| EST | Joueur en haut |
| SUD | Joueur Ã  gauche |
| OUEST | Joueur en bas |

Définition à la ligne 81 du fichier main.h.

#### 4.18.4.4 NbPoint

enum NbPoint

**Valeurs énumérées**

| | |
|---|---|
| ZERO | |
| QUATRE_VINGT | Le contrat vaut zÃ©ro point, par dÃ©faut correspont a un joueur qui passe ou pour l'initiatilation |
| QUATRE_VINGT_DIX | Le contrat vaut 80 points |
| CENT | Le contrat vaut 90 points |
| CENT_DIX | Le contrat vaut 100 points |
| CENT_VINGT | Le contrat vaut 110 points |
| CENT_TRENTE | Le contrat vaut 120 points |
| CENT_QUARANTE | Le contrat vaut 130 points |
| CENT_CINQUANTE | Le contrat vaut 140 points |
| CENT_SOIXANTE | Le contrat vaut 150 points |
| CAPOT | Le contrat vaut 160 points |
| GENERALE | Le contrat est un capot |

Définition à la ligne 112 du fichier main.h.

#### 4.18.4.5 TypeMessage

enum TypeMessage

contante Ã  passer en paramÃ¨tre Ã  la fonction gÃ©nÃ¨reMessage() pour donner le type de message a gÃ©nÃ©rer

**Valeurs énumérées**

| | |
|---|---|
| SANS_MESSAGE | |
| POSE_CARTE | pas de message a afficher, par defaut |
| RESULTAT_PLI | lorsqu'un joueur vien de poser une carte |
| RESULTAT_MANCHE | anonce le vainceur d'un plis |

Définition à la ligne 164 du fichier main.h.

#### 4.18.4.6 Valeur

enum Valeur

Les diffÃ©rentes valeur possible des cartes d'un jeu de 32 cartes.

Les valeurs associÃ©e on Ã©tÃ© placÃ©e de maniÃ¨re arbitraire et n'ont pas d'importance sur le programme Pour simplifier, le tout atout et le sans atout sont considÃ©rÃ© comme des couleurs

**Valeurs énumérées**

| | |
|---|---|
| SANS_VALEUR | |
| AS | Valeur par dÃ©faut, pour une carte vide |
| VALET | Valeur As |
| DAME | Valeur Valet |
| ROI | Valeur Dame |
| SEPT | Valeur Roi |
| HUIT | Valeur 7 |
| NEUF | Valeur 8 |
| DIX | Valeur 9 |

Définition à la ligne 60 du fichier main.h.

## 4.19 Référence du fichier sous-menus.c

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

Graphe des dépendances par inclusion de sous-menus.c:

## Fonctions

- void parametre (char ∗pseudo[ ])
  *affiche le sous menu des paramètre, permet de changer le psedo de tout les ordinateur*
- int leaderboard (FILE ∗fichier)
  *fonction qui recupère les meilleurs score dans un ficher et les affiche dans la console*
- int statistiqueJoueur (FILE ∗fichier, int ligne)
  *fonction qui recupère les statistiques du joueur et les affiches dans la console*

### 4.19.1 Documentation des fonctions

#### 4.19.1.1 leaderboard()

```
int leaderboard (
            FILE * fichier )
```

fonction qui recupère les meilleurs score dans un ficher et les affiche dans la console

**Paramètres**

| | |
|---|---|
| *pointeur* | vers le fichier à lire |

**Renvoie**

0

Définition à la ligne 32 du fichier sous-menus.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.19.1.2 parametre()

```
void parametre (
            char * pseudo[] )
```

affiche le sous menu des paramètre, permet de changer le psedo de tout les ordinateur

**Paramètres**

| | |
|---|---|
| *char* | ∗pseudo[] : tableau de pointeurs contenant les pseudos des différents joueurs |

**Renvoie**

void

Définition à la ligne 11 du fichier sous-menus.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.19.1.3 statistiqueJoueur()

```
int statistiqueJoueur (
            FILE * fichier,
            int ligne )
```

fonction qui recupère les statistiques du joueur et les affiches dans la console

**Paramètres**

| | |
|---|---|
| *pointeur* | vers le fichier de sauvegarde des scores |
| *numero* | de la ligne à laquelle sont ecrites les statistiques du joueur |

**Renvoie**

0

Définition à la ligne 85 du fichier sous-menus.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

## 4.20 Référence du fichier sous-menus.h

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

### Fonctions

- void parametre (char *pseudo[])
  *affiche le sous menu des paramètre, permet de changer le psedo de tout les ordinateur*
- int leaderboard (FILE *fichier)
  *fonction qui recupère les meilleurs score dans un ficher et les affiche dans la console*
- int statistiqueJoueur (FILE *fichier, int ligne)
  *fonction qui recupère les statistiques du joueur et les affiches dans la console*

### 4.20.1 Documentation des fonctions

#### 4.20.1.1 leaderboard()

```
int leaderboard (
            FILE * fichier )
```

fonction qui recupère les meilleurs score dans un ficher et les affiche dans la console

**Paramètres**

| | |
|---|---|
| *pointeur* | vers le fichier à lire |

**Renvoie**

0

Définition à la ligne 32 du fichier sous-menus.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.20.1.2 parametre()

```
void parametre (
            char * pseudo[] )
```

affiche le sous menu des paramètre, permet de changer le psedo de tout les ordinateur

**Paramètres**

| | |
|---|---|
| *char* | ∗pseudo[] : tableau de pointeurs contenant les pseudos des différents joueurs |

**Renvoie**

void

Définition à la ligne 11 du fichier sous-menus.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.20.1.3 statistiqueJoueur()

```
int statistiqueJoueur (
            FILE * fichier,
            int ligne )
```

fonction qui recupère les statistiques du joueur et les affiches dans la console

**Paramètres**

| | |
|---|---|
| *pointeur* | vers le fichier de sauvegarde des scores |
| *numero* | de la ligne à laquelle sont ecrites les statistiques du joueur |

**Renvoie**

0

Définition à la ligne 85 du fichier sous-menus.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

## 4.21 Référence du fichier tableau.c

ficher source contenant les fonctions relative à la gestion des tableau

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

Graphe des dépendances par inclusion de tableau.c:

### Fonctions

- int sommeTableau (int tableau[ ], int tailleTableau)
  *fait la somme des valeur d'un tableau contenant des entiers*
- float moyenneTableau (int tableau[ ], int tailleTableau)
  *fait la moyenne des valeur d'un tableau contenant des entiers*
- void copie (int tableauOriginal[ ], int tableauCopie[ ], int tailleTableau)
  *copie un tableau dans un autre*
- int mini (int tableau[ ], int tailleTableau)
  *trouve la valeur minimum d'un tableau*
- int maxi (int tableau[ ], int tailleTableau)
  *trouve la valeur maximum d'un tableau*
- void zeroSiSuperieur (int tableau[ ], int tailleTableau, int maximum)
  *met a 0 toute les valeur d'un tableau qui sont superieur a un maximum*
- void ordonnerTableau (int tableau[ ], int tailleTableau)
  *classe les valeur d'un tableau dans l'orde croissant*
- void afficheTableau (int tableau[ ], int tailleTableau)
  *affiche un tableau*
- void acquiertTableau (int tableau[ ], int tailleTableau)
  *demande a l'utilisateurde saisir les valeurs d'un tableau*
- void constanteTableau (int tableau[ ], int tailleTableau, int valeur)
  *donne a toute les valeur d'un tableau une constante*
- int rechercheDichotomie (int tableau[ ], int tailleTableau, int nCherche)
  *chercherche par dicotomie si une valeur est au moin une fois dans le tableau*
- void aleatoireTableau (int tableau[ ], int tailleTableau, int mini, int maxi)
  *donne des nombres aléatoire au valeur d'un tableau*

### 4.21.1 Description détaillée

ficher source contenant les fonctions relative à la gestion des tableau

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.21.2 Documentation des fonctions

#### 4.21.2.1 acquiertTableau()

```
void acquiertTableau (
            int tableau[],
            int tailleTableau )
```

demande a l'utilisateurde saisir les valeurs d'un tableau

**Paramètres**

| | |
|---|---|
| *tableau[]* | : nom du tableau a saisir |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

void

Définition à la ligne 93 du fichier tableau.c.

#### 4.21.2.2 afficheTableau()

```
void afficheTableau (
            int tableau[],
            int tailleTableau )
```

affiche un tableau

**Paramètres**

| | |
|---|---|
| *tableau[]* | : le tableau a afficher |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

void

Définition à la ligne 83 du fichier tableau.c.

#### 4.21.2.3 aleatoireTableau()

```
void aleatoireTableau (
            int tableau[],
            int tailleTableau,
            int mini,
            int maxi )
```

donne des nombres aléatoire au valeur d'un tableau

**Paramètres**

| | |
|---|---|
| *tableau[]* | : tableau a modifier |
| *tailleTableau* | : taille du tableau entier positif |
| *mini* | : valeur minimum des nombres aléatoires |
| *maxi* | : valeur maximum des nombres aléatoires |

**Renvoie**

Définition à la ligne 127 du fichier tableau.c.

Voici le graphe d'appel pour cette fonction :

#### 4.21.2.4 constanteTableau()

```
void constanteTableau (
             int tableau[],
             int tailleTableau,
             int valeur )
```

donne a toute les valeur d'un tableau une constante

**Paramètres**

| | |
|---|---|
| *tableau[]* | : tableau a modifier |
| *tailleTableau* | : taille du tableau entier positif |
| *valeur* | : valeur a donner a toutes les valeur du tableau |

**Renvoie**

void

Définition à la ligne 101 du fichier tableau.c.

#### 4.21.2.5 copie()

```
void copie (
             int tableauOriginal[],
             int tableauCopie[],
             int tailleTableau )
```

copie un tableau dans un autre

**Paramètres**

| | |
|---|---|
| *tableauOriginal[]* | : tableaau que l'on veut copier |
| *tableauCopie[]* | : tableau dans lequel on copie tableauOriaginal |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

void

Définition à la ligne 25 du fichier tableau.c.

#### 4.21.2.6 maxi()

```
int maxi (
             int tableau[],
             int tailleTableau )
```

trouve la valeur maximum d'un tableau

**Paramètres**

| | |
|---|---|
| *tableau* | [] : tableau dont on veut travailler |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

int valeur la plus grande du tableau

Définition à la ligne 45 du fichier tableau.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.21.2.7 mini()

```
int mini (
             int tableau[],
             int tailleTableau )
```

trouve la valeur minimum d'un tableau

**Paramètres**

| | |
|---|---|
| *tableau* | [] : tableau dont on veut travailler |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

int valeur la plus petite du tableau

Définition à la ligne 33 du fichier tableau.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.21.2.8 moyenneTableau()

```
float moyenneTableau (
             int tableau[],
             int tailleTableau )
```

fait la moyenne des valeur d'un tableau contenant des entiers

**Paramètres**

| | |
|---|---|
| *tableau[]* | : le nom du tableau |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

la moyenne des valeur du tableau, réel

Définition à la ligne 20 du fichier tableau.c.

Voici le graphe d'appel pour cette fonction :

#### 4.21.2.9 ordonnerTableau()

```
void ordonnerTableau (
            int tableau[],
            int tailleTableau )
```

classe les valeur d'un tableau dans l'orde croissant

**Paramètres**

| | |
|---|---|
| *tableau[]* | : le tableau à classer |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

void

Définition à la ligne 66 du fichier tableau.c.

#### 4.21.2.10 rechercheDichotomie()

```
int rechercheDichotomie (
            int tableau[],
            int tailleTableau,
            int nCherche )
```

chercherche par dicotomie si une valeur est au moin une fois dans le tableau

**Paramètres**

| | |
|---|---|
| *tableau[]* | : tableau dans lequel on fait la recherche |
| *tailleTableau* | : taille du tableau entier positif |
| *nCherche* | nombre que l'on cherche dans le tableau |

**Renvoie**

int : 1 si on a trouvé, 0 sinon

Définition à la ligne 108 du fichier tableau.c.

#### 4.21.2.11 sommeTableau()

```
int sommeTableau (
            int tableau[],
            int tailleTableau )
```

fait la somme des valeur d'un tableau contenant des entiers

**Paramètres**

| | |
|---|---|
| *tableau[]* | : le nom du tableau |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

la somme des valeur du table entier

Définition à la ligne 11 du fichier tableau.c.

Voici le graphe des appelants de cette fonction :

#### 4.21.2.12 zeroSiSuperieur()

```
void zeroSiSuperieur (
            int tableau[],
            int tailleTableau,
            int maximum )
```

met a 0 toute les valeur d'un tableau qui sont superieur a un maximum

**Paramètres**

| | |
|---|---|
| *tableau[]* | : tableau sur lequel on veut travailler |
| *tailleTableau* | : la taille du tableau, entier positif |
| *maximum* | : valeur maximum au dessus de quoi la valeur est remise a 0 |

**Renvoie**

void

Définition à la ligne 57 du fichier tableau.c.

# 4.22 Référence du fichier tableau.h

ficher header contenant les prototypes des fonctions relative à la gestion des tableau

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

## Fonctions

- int sommeTableau (int tableau[ ], int tailleTableau)
  *fait la somme des valeur d'un tableau contenant des entiers*
- float moyenneTableau (int tableau[ ], int tailleTableau)
  *fait la moyenne des valeur d'un tableau contenant des entiers*
- void copie (int tableauOriginal[ ], int tableauCopie[ ], int tailleTableau)
  *copie un tableau dans un autre*
- int mini (int tableau[ ], int tailleTableau)
  *trouve la valeur minimum d'un tableau*
- int maxi (int tableau[ ], int tailleTableau)
  *trouve la valeur maximum d'un tableau*
- void zeroSiSuperieur (int tableau[ ], int tailleTableau, int maximum)
  *met a 0 toute les valeur d'un tableau qui sont superieur a un maximum*
- void ordonnerTableau (int tableau[ ], int tailleTableau)
  *classe les valeur d'un tableau dans l'orde croissant*
- void afficheTableau (int tableau[ ], int tailleTableau)
  *affiche un tableau*
- void acquiertTableau (int tableau[ ], int tailleTableau)
  *demande a l'utilisateurde saisir les valeurs d'un tableau*
- void constanteTableau (int tableau[ ], int tailleTableau, int valeur)
  *donne a toute les valeur d'un tableau une constante*
- void aleatoireTableau (int tableau[ ], int tailleTableau, int mini, int maxi)
  *donne des nombres aléatoire au valeur d'un tableau*
- int rechercheDichotomie (int tableau[ ], int tailleTableau, int nCherche)
  *chercherche par dicotomie si une valeur est au moin une fois dans le tableau*

### 4.22.1 Description détaillée

ficher header contenant les prototypes des fonctions relative à la gestion des tableau

**Auteur**

Carlo.A & Florian.C

**Version**

v1.0

**Date**

12 juin 2020

### 4.22.2 Documentation des fonctions

#### 4.22.2.1 acquiertTableau()

```
void acquiertTableau (
            int tableau[],
            int tailleTableau )
```

demande a l'utilisateurde saisir les valeurs d'un tableau

**Paramètres**

| | |
|---|---|
| *tableau[]* | : nom du tableau a saisir |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

void

Définition à la ligne 93 du fichier tableau.c.

#### 4.22.2.2 afficheTableau()

```
void afficheTableau (
            int tableau[],
            int tailleTableau )
```

affiche un tableau

**Paramètres**

| | |
|---|---|
| *tableau[]* | : le tableau a afficher |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

void

Définition à la ligne 83 du fichier tableau.c.

#### 4.22.2.3 aleatoireTableau()

```
void aleatoireTableau (
            int tableau[],
            int tailleTableau,
            int mini,
            int maxi )
```

donne des nombres aléatoire au valeur d'un tableau

**Paramètres**

| | |
|---|---|
| *tableau[]* | : tableau a modifier |
| *tailleTableau* | : taille du tableau entier positif |
| *mini* | : valeur minimum des nombres aléatoires |
| *maxi* | : valeur maximum des nombres aléatoires |

**Renvoie**

Définition à la ligne 127 du fichier tableau.c.

Voici le graphe d'appel pour cette fonction :

#### 4.22.2.4 constanteTableau()

```
void constanteTableau (
             int tableau[],
             int tailleTableau,
             int valeur )
```

donne a toute les valeur d'un tableau une constante

**Paramètres**

| | |
|---|---|
| *tableau[]* | : tableau a modifier |
| *tailleTableau* | : taille du tableau entier positif |
| *valeur* | : valeur a donner a toutes les valeur du tableau |

**Renvoie**

void

Définition à la ligne 101 du fichier tableau.c.

#### 4.22.2.5 copie()

```
void copie (
             int tableauOriginal[],
             int tableauCopie[],
             int tailleTableau )
```

copie un tableau dans un autre

**Paramètres**

| | |
|---|---|
| *tableauOriginal[]* | : tableaau que l'on veut copier |
| *tableauCopie[]* | : tableau dans lequel on copie tableauOriaginal |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

void

Définition à la ligne 25 du fichier tableau.c.

#### 4.22.2.6 maxi()

```
int maxi (
             int tableau[],
             int tailleTableau )
```

trouve la valeur maximum d'un tableau

**Paramètres**

| | |
|---|---|
| *tableau* | [] : tableau dont on veut travailler |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

int valeur la plus grande du tableau

Définition à la ligne 45 du fichier tableau.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.22.2.7 mini()

```
int mini (
             int tableau[],
             int tailleTableau )
```

trouve la valeur minimum d'un tableau

**Paramètres**

| | |
|---|---|
| *tableau* | [] : tableau dont on veut travailler |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

int valeur la plus petite du tableau

Définition à la ligne 33 du fichier tableau.c.

Voici le graphe d'appel pour cette fonction : Voici le graphe des appelants de cette fonction :

#### 4.22.2.8 moyenneTableau()

```
float moyenneTableau (
             int tableau[],
             int tailleTableau )
```

fait la moyenne des valeur d'un tableau contenant des entiers

**Paramètres**

| | |
|---|---|
| *tableau[]* | : le nom du tableau |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

la moyenne des valeur du tableau, réel

Définition à la ligne 20 du fichier tableau.c.

Voici le graphe d'appel pour cette fonction :

#### 4.22.2.9 ordonnerTableau()

```
void ordonnerTableau (
            int tableau[],
            int tailleTableau )
```

classe les valeur d'un tableau dans l'orde croissant

**Paramètres**

| | |
|---|---|
| *tableau[]* | : le tableau à classer |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

void

Définition à la ligne 66 du fichier tableau.c.

#### 4.22.2.10 rechercheDichotomie()

```
int rechercheDichotomie (
            int tableau[],
            int tailleTableau,
            int nCherche )
```

chercherche par dicotomie si une valeur est au moin une fois dans le tableau

**Paramètres**

| | |
|---|---|
| *tableau[]* | : tableau dans lequel on fait la recherche |
| *tailleTableau* | : taille du tableau entier positif |
| *nCherche* | nombre que l'on cherche dans le tableau |

**Renvoie**

int : 1 si on a trouvé, 0 sinon

Définition à la ligne 108 du fichier tableau.c.

#### 4.22.2.11 sommeTableau()

```
int sommeTableau (
            int tableau[],
            int tailleTableau )
```

fait la somme des valeur d'un tableau contenant des entiers

**Paramètres**

| | |
|---|---|
| *tableau[]* | : le nom du tableau |
| *tailleTableau* | : la taille du tableau, entier positif |

**Renvoie**

la somme des valeur du table entier

Définition à la ligne 11 du fichier tableau.c.

Voici le graphe des appelants de cette fonction :

#### 4.22.2.12 zeroSiSuperieur()

```
void zeroSiSuperieur (
            int tableau[],
            int tailleTableau,
            int maximum )
```

met a 0 toute les valeur d'un tableau qui sont superieur a un maximum

**Paramètres**

| | |
|---|---|
| *tableau[]* | : tableau sur lequel on veut travailler |
| *tailleTableau* | : la taille du tableau, entier positif |
| *maximum* | : valeur maximum au dessus de quoi la valeur est remise a 0 |

**Renvoie**

void

Définition à la ligne 57 du fichier tableau.c.

# Index