Coinche Project Documentation

Generated by Doxygen 1.8.18

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 Card Struct Reference

`#include <core.h>`

Data Fields

- Value value
- Color color
- Bool canPlay

3.1.1 Field Documentation

3.1.1.1 canPlay

`Bool canPlay`

3.1.1.2 color

`Color color`

3.1.1.3 value

`Value value`

The documentation for this struct was generated from the following file:

- F:/C_IFE/IFE-coinche/src/core.h

3.2 Contract Struct Reference

`#include <core.h>`

Data Fields

- Color trump
- ContractType type
- int points
- Coinche coinche
- Position issuer

3.2.1 Field Documentation

3.2.1.1 coinche

`Coinche coinche`

3.2.1.2 issuer

`Position issuer`

3.2.1.3 points

`int points`

3.2.1.4 trump

`Color trump`

3.2.1.5 type

`ContractType type`

The documentation for this struct was generated from the following file:

- F:/C_IFE/IFE-coinche/src/core.h

3.3 Player Struct Reference

`#include <core.h>`

Collaboration diagram for Player:

Data Fields

- CardAI cardAI
- ContractAI contractAI
- Position pos
- char * name
- char * croppedName
- Card * cards
- int nbOfCards
- int score
- int teamScore

3.3.1 Field Documentation

3.3.1.1 cardAI

```
CardAI cardAI
```

3.3.1.2 cards

```
Card* cards
```

3.3.1.3 contractAI

```
ContractAI contractAI
```

3.3.1.4 croppedName

```
char* croppedName
```

3.3.1.5 name

```
char* name
```

3.3.1.6 nbOfCards

```
int nbOfCards
```

3.3.1.7 pos

```
Position pos
```

3.3.1.8 score

```
int score
```

3.3.1.9 teamScore

```
int teamScore
```

The documentation for this struct was generated from the following file:

- F:/C_IFE/IFE-coinche/src/core.h

Chapter 4

File Documentation

4.1 F:/C_IFE/IFE-coinche/src/ai.c File Reference

```
#include "ai.h"
#include "cardUtils.h"
```

Include dependency graph for ai.c:

4.2 F:/C_IFE/IFE-coinche/src/ai.h File Reference

```
#include "core.h"
```

Include dependency graph for ai.h: This graph shows which files directly or indirectly include this file:

Functions

- Card getAICardFirstAvailable (Card cardsInHand[ ], int nbOfCardsInHand)
- Card getAICardStandard (Card cardsInHand[ ], int nbOfCardsInHand, Card trickCards[ ], int nbOfTrickCards, Color trump, Color roundColor)
- Bool getAIContractAlwaysEighty (Card cardsInHand[ ], Contract ∗contract)
- Bool getAIContractStandard (Card cardsInHand[ ], int nbOfCardsInHand, Contract ∗contract)

4.2.1 Function Documentation

4.2.1.1 getAICardFirstAvailable()

```
Card getAICardFirstAvailable (
            Card cardsInHand[],
            int nbOfCardsInHand )
```

Parameters

cardsInHand[]	array containing the cards in the AI's hand
nbOfCardsInHand	the number of cards in the array cardsInHand

Returns

chosenCard: the chosen card

Returns the first playable card in the AI's hand Here is the caller graph for this function:

4.2.1.2 getAICardStandard()

```
Card getAICardStandard (
            Card cardsInHand[],
            int nbOfCardsInHand,
            Card trickCards[],
            int nbOfTrickCards,
            Color trump,
            Color roundColor )
```

Parameters

cardsInHand[]	array containing the cards in the AI's hand
nbOfCardsInHand	the number of cards in in the array cardsInHand
trickCards[]	array containing the cards already played by the previous players. Only the first N cards matter, where N is nbOfTrickCards
nbOfTrickCards	the number of cards played by the previous players. Can be 0
trump	the current trump
roundColor	the color of the first card played of that trick

Returns

chosenCard: the chosen card

If the AI is able to win the trick with one of its cards, it does so by playing the lowest possible card If the AI is unable to win, it plays its lowest card Here is the call graph for this function: Here is the caller graph for this function:

4.2.1.3 getAIContractAlwaysEighty()

```
Bool getAIContractAlwaysEighty (
            Card cardsInHand[],
            Contract * contract )
```

Parameters

cardsInHand[]	array containing the cards in the AI's hand
∗contract	pointer to the contract being debated. Will be edited if the AI decides to make a contract

Returns

hasPassed: TRUE if the AI has decided to pass, FALSE if it decided to make a contract

This AI always makes an 80 points contract in its first card's color if it can
Here is the caller graph for this function:

4.2.1.4 getAIContractStandard()

```
Bool getAIContractStandard (
            Card cardsInHand[],
            int nbOfCardsInHand,
            Contract * contract )
```

Parameters

cardsInHand[]	array containing the cards in the AI's hand
nbOfCardsInHand	the number of cards in hand
∗contract	pointer to the contract being debated. Will be edited if the AI decides to make a contract

Returns

hasPassed: TRUE if the AI has decided to pass, FALSE if it decided to make a contract

If the AI has 4 strong cards or more in a given color, it makes a 120 points contract. If it has 3 strong cards, it makes an 80 points one Here is the call graph for this function: Here is the caller graph for this function:

4.3 F:/C_IFE/IFE-coinche/src/cardUtils.c File Reference

```
#include "cardUtils.h"
```
Include dependency graph for cardUtils.c:

Functions

- int getCardStrength (Card card, Color trump, Color roundColor)
- int getStrongestCard (Card cardArray[ ], int nbOfCards, Color trump, Color roundColor)
- int getCardPoints (Card card, Color trump)
- int getCardArrayPoints (Card cardArray[ ], int nbOfCards, Color trump)
- Bool setCanPlay (Card cardArray[ ], int nbOfCards, Color conditionalColor, Color trump, int bestTrump↩ Strength, Bool canPlay)
- void findValidCardsInHand (Card cardsInHand[ ], int nbOfCardsInHand, Card trickCards[ ], int nbOfTrick↩ Cards, Color trump)
- void sortCards (Card cardArray[ ], int nbToSort, Color trump, Color roundColor)
- Bool removeCard (Card cardArray[ ], int ∗nbOfCards, Card cardToRemove)
- int getPlayableCards (Card cardArray[ ], int nbOfCards, Card playableCards[ ])
- void createDeck (Card cardDeck[ ])

4.3.1 Function Documentation

4.3.1.1 createDeck()

```
void createDeck (
            Card cardDeck[] )
```

Parameters

cardDeck[]	must be a 32 long array. Will be filled with the deck

Creation of the deck containing all 32 cards Here is the caller graph for this function:

4.3.1.2 findValidCardsInHand()

```
void findValidCardsInHand (
            Card cardsInHand[],
            int nbOfCardsInHand,
            Card trickCards[],
            int nbOfTrickCards,
            Color trump )
```

Parameters

cardsInHand[]	the cards in this array will be evaluated to find the valid ones to play according to the coinche rules
nbOfCardsInHand	the number of cards this function should look at to see which ones are valid
trickCards[]	the cards already played by the previous players. Only the first N cards matter, where N is nbOfTrickCards
nbOfTrickCards	the number of cards played by the previous players. Can be 0
trump	the current trump

Sets the canPlay property of each card in an Array to find which ones can be played depending on the cards currently on the table Here is the call graph for this function: Here is the caller graph for this function:

4.3.1.3 getCardArrayPoints()

```
int getCardArrayPoints (
            Card cardArray[],
            int nbOfCards,
            Color trump )
```

Parameters

cardArray[]	array containing the cards
nbOfCards	how many cards are in cardArray
trump	the current trump

Returns

totalPoints -> the sum of the point value of each card in cardArray

Computes the total point value of an Array of cards Here is the call graph for this function: Here is the caller graph for this function:

4.3.1.4 getCardPoints()

```
int getCardPoints (
            Card card,
            Color trump )
```

Parameters

card	the evaluated card
trump	the current trump

Returns

cardPoints -> the point value of the card

Finds the point value of a card depending on the current trump Here is the caller graph for this function:

4.3.1.5 getCardStrength()

```
int getCardStrength (
            Card card,
            Color trump,
            Color roundColor )
```

Parameters

card	the evaluated card
trump	the current trump
roundColor	the color of the first played card in the round

Returns

cardStrength -> an integer that defines the "strength" of the card, where the card with the greatest "strength" wins the trick

Defines the "strength" of a played card in order to compare cards and find which one is stronger Here is the caller graph for this function:

4.3.1.6 getPlayableCards()

```
int getPlayableCards (
            Card cardArray[],
            int nbOfCards,
            Card playableCards[] )
```

Parameters

cardArray[]	array containing the cards
nbOfCards	the number of cards in cardArray
playableCards[]	array to store the playableCards in. This array must be big enough to store all playable cards

Returns

nbOfPlayableCards: the number of cards that ended up in playableCards

In an array of cards, return only those with canPlay = TRUE, the function findValidCardsInHand must have been called before Here is the caller graph for this function:

4.3.1.7 getStrongestCard()

```
int getStrongestCard (
            Card cardArray[],
            int nbOfCards,
            Color trump,
            Color roundColor )
```

Parameters

cardArray[]	array containing the cards to compare
nbOfCards	how many cards are being compared. Can be a single card
trump	the current trump
roundColor	the color of the first played card in the round

Returns

strongestCardPos -> the position of the strongest card in the set, where 0 is the first card of the Array

Finds the strongest card in a set Here is the call graph for this function: Here is the caller graph for this function:

4.3.1.8 removeCard()

```
Bool removeCard (
            Card cardArray[],
            int * nbOfCards,
            Card cardToRemove )
```

Parameters

cardArray[]	array containing the cards
∗nbOfCards	pointer to the number of cards in cardArray
cardToRemove	the card to seek and remove in cardArray (it's canPlay state doesn't matter)

Returns

foundCard: TRUE if the function was able to find the card (thus removing it), FALSE otherwise

Seeks a card in an array to remove it, and decreases the number of cards if it removes one Here is the caller graph for this function:

4.3.1.9 setCanPlay()

```
Bool setCanPlay (
            Card cardArray[],
            int nbOfCards,
            Color conditionalColor,
            Color trump,
            int bestTrumpStrength,
            Bool canPlay )
```

Parameters

cardArray[]	array containing the cards
nbOfCards	how many cards are in cardArray
conditionalColor	the canPlay property of each card will be set if its color is conditionalColor. Set to NULL_COLOR to bypass the condition
trump	the current trump. Note: this cannot be ALLTRUMP. If it is, replace ALLTRUMP with the current trick color
bestTrumpStrength	the current best trump's Strength. A card will not be set if it's a trump weaker than this. Set to 0 to bypass the condition
canPlay	the Bool value canPlay property should be set to

Returns

conditionMet -> FALSE if no canPlay property was changed, TRUE otherwise

Sets the canPlay property of each card if it meets a color criteria and if it's not a weaker trump than the current best one
Here is the call graph for this function: Here is the caller graph for this function:

4.3.1.10 sortCards()

```
void sortCards (
            Card cardArray[],
            int nbToSort,
            Color trump,
            Color roundColor )
```

Parameters

cardArray[]	array containing the cards to sort
nbToSort	the first N cards of the array that will be sorted. Usually set to the array length to sort the whole array
trump	the current trump
roundColor	the color of the first played card in the round

Sorts the cards in an array from weakest to strongest Here is the call graph for this function: Here is the caller graph for this function:

4.4 F:/C_IFE/IFE-coinche/src/cardUtils.h File Reference

`#include "core.h"`
Include dependency graph for cardUtils.h: This graph shows which files directly or indirectly include this file:

Functions

- int getCardStrength (Card card, Color trump, Color roundColor)
- int getStrongestCard (Card cardArray[ ], int nbOfCards, Color trump, Color roundColor)
- int getCardPoints (Card card, Color trump)
- int getCardArrayPoints (Card cardArray[ ], int nbOfCards, Color trump)
- Bool setCanPlay (Card cardArray[ ], int nbOfCards, Color conditionalColor, Color trump, int bestTrump↩ Strength, Bool canPlay)
- void findValidCardsInHand (Card cardsInHand[ ], int nbOfCardsInHand, Card trickCards[ ], int nbOfTrick↩ Cards, Color trump)
- void sortCards (Card cardArray[ ], int nbToSort, Color trump, Color roundColor)
- Bool removeCard (Card cardArray[ ], int ∗nbOfCards, Card cardToRemove)
- int getPlayableCards (Card cardArray[ ], int nbOfCards, Card playableCards[ ])
- void createDeck (Card cardDeck[ ])

4.4.1 Function Documentation

4.4.1.1 createDeck()

```
void createDeck (
            Card cardDeck[ ] )
```

Parameters

cardDeck[ ]	must be a 32 long array. Will be filled with the deck

Creation of the deck containing all 32 cards Here is the caller graph for this function:

4.4.1.2 findValidCardsInHand()

```
void findValidCardsInHand (
            Card cardsInHand[],
            int nbOfCardsInHand,
            Card trickCards[],
            int nbOfTrickCards,
            Color trump )
```

Parameters

cardsInHand[]	the cards in this array will be evaluated to find the valid ones to play according to the coinche rules
nbOfCardsInHand	the number of cards this function should look at to see which ones are valid
trickCards[]	the cards already played by the previous players. Only the first N cards matter, where N is nbOfTrickCards
nbOfTrickCards	the number of cards played by the previous players. Can be 0
trump	the current trump

Sets the canPlay property of each card in an Array to find which ones can be played depending on the cards currently on the table Here is the call graph for this function: Here is the caller graph for this function:

4.4.1.3 getCardArrayPoints()

```
int getCardArrayPoints (
            Card cardArray[],
            int nbOfCards,
            Color trump )
```

Parameters

cardArray[]	array containing the cards
nbOfCards	how many cards are in cardArray
trump	the current trump

Returns

totalPoints -> the sum of the point value of each card in cardArray

Computes the total point value of an Array of cards Here is the call graph for this function: Here is the caller graph for this function:

4.4.1.4 getCardPoints()

```
int getCardPoints (
            Card card,
            Color trump )
```

Parameters

card	the evaluated card
trump	the current trump

Returns

cardPoints -> the point value of the card

Finds the point value of a card depending on the current trump Here is the caller graph for this function:

4.4.1.5 getCardStrength()

```
int getCardStrength (
            Card card,
            Color trump,
            Color roundColor )
```

Parameters

card	the evaluated card
trump	the current trump
roundColor	the color of the first played card in the round

Returns

cardStrength -> an integer that defines the "strength" of the card, where the card with the greatest "strength" wins the trick

Defines the "strength" of a played card in order to compare cards and find which one is stronger Here is the caller graph for this function:

4.4.1.6 getPlayableCards()

```
int getPlayableCards (
            Card cardArray[],
            int nbOfCards,
            Card playableCards[] )
```

Parameters

cardArray[]	array containing the cards
nbOfCards	the number of cards in cardArray
playableCards[]	array to store the playableCards in. This array must be big enough to store all playable cards

Returns

nbOfPlayableCards: the number of cards that ended up in playableCards

In an array of cards, return only those with canPlay = TRUE, the function findValidCardsInHand must have been called before Here is the caller graph for this function:

4.4.1.7 getStrongestCard()

```
int getStrongestCard (
            Card cardArray[],
            int nbOfCards,
            Color trump,
            Color roundColor )
```

Parameters

cardArray[]	array containing the cards to compare
nbOfCards	how many cards are being compared. Can be a single card
trump	the current trump
roundColor	the color of the first played card in the round

Returns

strongestCardPos -> the position of the strongest card in the set, where 0 is the first card of the Array

Finds the strongest card in a set Here is the call graph for this function: Here is the caller graph for this function:

4.4.1.8 removeCard()

```
Bool removeCard (
            Card cardArray[],
            int * nbOfCards,
            Card cardToRemove )
```

Parameters

cardArray[]	array containing the cards
∗nbOfCards	pointer to the number of cards in cardArray
cardToRemove	the card to seek and remove in cardArray (it's canPlay state doesn't matter)

Returns

foundCard: TRUE if the function was able to find the card (thus removing it), FALSE otherwise

Seeks a card in an array to remove it, and decreases the number of cards if it removes one Here is the caller graph for this function:

4.4.1.9 setCanPlay()

```
Bool setCanPlay (
            Card cardArray[],
            int nbOfCards,
            Color conditionalColor,
            Color trump,
            int bestTrumpStrength,
            Bool canPlay )
```

Parameters

cardArray[]	array containing the cards
nbOfCards	how many cards are in cardArray
conditionalColor	the canPlay property of each card will be set if its color is conditionalColor. Set to NULL_COLOR to bypass the condition

Parameters

trump	the current trump. Note: this cannot be ALLTRUMP. If it is, replace ALLTRUMP with the current trick color
bestTrumpStrength	the current best trump's Strength. A card will not be set if it's a trump weaker than this. Set to 0 to bypass the condition
canPlay	the Bool value canPlay property should be set to

Returns

conditionMet -> FALSE if no canPlay property was changed, TRUE otherwise

Sets the canPlay property of each card if it meets a color criteria and if it's not a weaker trump than the current best one
Here is the call graph for this function: Here is the caller graph for this function:

4.4.1.10 sortCards()

```
void sortCards (
            Card cardArray[],
            int nbToSort,
            Color trump,
            Color roundColor )
```

Parameters

cardArray[]	array containing the cards to sort
nbToSort	the first N cards of the array that will be sorted. Usually set to the array length to sort the whole array
trump	the current trump
roundColor	the color of the first played card in the round

Sorts the cards in an array from weakest to strongest Here is the call graph for this function: Here is the caller graph for this function:

4.5 F:/C_IFE/IFE-coinche/src/core.c File Reference

```
#include "core.h"
```
Include dependency graph for core.c:

Variables

- const int CARD_POINTS_TABLE [4][8]
- const char * VALUE_STR_TABLE [9][2]
- const char * COLOR_STR_TABLE [7][2]
- const char * COINCHE_STR_TABLE [3]
- const char * CONTRACTTYPE_STR_TABLE [3]
- const char * CARDAI_STR_TABLE [NB_CARD_AI]
- const char * CONTRACTAI_STR_TABLE [NB_CONTRACT_AI]

4.5.1 Variable Documentation

4.5.1.1 CARD_POINTS_TABLE

```
const int CARD_POINTS_TABLE[4][8]
```

Initial value:

```
= {
    {    0,     0,     9,    14,     1,     3,     5,     6},
    {    0,     0,     0,     2,     3,     4,    10,    19},
    {    0,     0,    14,    20,     3,     4,    10,    11},
    {    0,     0,     0,     2,     3,     4,    10,    11},
}
```

4.5.1.2 CARDAI_STR_TABLE

```
const char* CARDAI_STR_TABLE[NB_CARD_AI]
```

Initial value:

```
= {
    "User",
    "First available",
    "Standard"
}
```

4.5.1.3 COINCHE_STR_TABLE

```
const char* COINCHE_STR_TABLE[3]
```

Initial value:

```
= {
    "",
    "Coinched",
    "Overcoinched"
}
```

4.5.1.4 COLOR_STR_TABLE

```
const char* COLOR_STR_TABLE[7][2]
```

Initial value:

```
= {
    {"", ""},
    {"", " spade"},
    {"\033[0;31m\033[0m", " heart"},
    {"\033[0;31m\033[0m", " diamond"},
    {"", " club"},
    {"All trump", "n all trump"},
    {"No trump", "no trump"}
}
```

4.5.1.5 CONTRACTAI_STR_TABLE

```
const char* CONTRACTAI_STR_TABLE[NB_CONTRACT_AI]
```

Initial value:

```
= {
    "User",
    "Always eighty",
    "Standard"
}
```

4.5.1.6 CONTRACTTYPE_STR_TABLE

```
const char* CONTRACTTYPE_STR_TABLE[3]
```

Initial value:

```
= {
    "",
    "Capot",
    "General"
}
```

4.5.1.7 VALUE_STR_TABLE

```
const char* VALUE_STR_TABLE[9][2]
```

Initial value:

```
= {
    {"", ""},
    {"7 ", "7"},
    {"8 ", "8"},
    {"9 ", "9"},
    {"J ", "J"},
    {"Q ", "Q"},
    {"K ", "K"},
    {"10", "10"},
    {"A ", "A"}
}
```

4.6 F:/C_IFE/IFE-coinche/src/core.h File Reference

This graph shows which files directly or indirectly include this file:

Data Structures

- struct Card
- struct Player
- struct Contract

Macros

- #define MAX_PLAYER_NAME_LENGTH 50
- #define NB_CARD_AI 3
- #define NB_CONTRACT_AI 3

Typedefs

- typedef enum Bool Bool
- typedef enum Value Value
- typedef enum Color Color
- typedef struct Card Card
- typedef enum CardAI CardAI
- typedef enum ContractAI ContractAI
- typedef enum Position Position
- typedef struct Player Player
- typedef enum Coinche Coinche
- typedef enum ContractType ContractType
- typedef struct Contract Contract
- typedef enum TextPosition TextPosition

Enumerations

- enum Bool { FALSE = 0, TRUE = 1 }
- enum Value {
  NULL_VALUE = 0, SEVEN = 1, EIGHT = 2, NINE = 3,
  JACK = 4, QUEEN = 5, KING = 6, TEN = 7,
  ACE = 8 }
- enum Color {
  NULL_COLOR = 0, SPADE = 1, HEART = 2, DIAMOND = 3,
  CLUB = 4, ALLTRUMP = 5, NOTRUMP = 6 }
- enum CardAI { CARD_USER = 0, CARD_AI_FIRSTAVAILABLE = 1, CARD_AI_STANDARD = 2 }
- enum ContractAI { CONTRACT_USER = 0, CONTRACT_AI_ALWAYSEIGHTY = 1, CONTRACT_AI_STANDARD = 2 }
- enum Position { SOUTH = 0, WEST = 1, NORTH = 2, EAST = 3 }
- enum Coinche { NOT_COINCHED = 0, COINCHED = 1, OVERCOINCHED = 2 }
- enum ContractType { POINTS = 0, CAPOT = 1, GENERAL = 2 }
- enum TextPosition { TEXT_LEFT = 0, TEXT_CENTER = 1, TEXT_RIGHT = 2 }

Variables

- const int CARD_POINTS_TABLE [4][8]
- const char * VALUE_STR_TABLE [9][2]
- const char * COLOR_STR_TABLE [7][2]
- const char * COINCHE_STR_TABLE [3]
- const char * CONTRACTTYPE_STR_TABLE [3]
- const char * CARDAI_STR_TABLE [NB_CARD_AI]
- const char * CONTRACTAI_STR_TABLE [NB_CONTRACT_AI]

4.6.1 Macro Definition Documentation

4.6.1.1 MAX_PLAYER_NAME_LENGTH

```
#define MAX_PLAYER_NAME_LENGTH 50
```

4.6.1.2 NB_CARD_AI

```
#define NB_CARD_AI 3
```

4.6.1.3 NB_CONTRACT_AI

```
#define NB_CONTRACT_AI 3
```

4.6.2 Typedef Documentation

4.6.2.1 Bool

```
typedef enum Bool Bool
```

4.6.2.2 Card

```
typedef struct Card Card
```

4.6.2.3 CardAI

```
typedef enum CardAI CardAI
```

4.6.2.4 Coinche

```
typedef enum Coinche Coinche
```

4.6.2.5 Color

```
typedef enum Color Color
```

4.6.2.6 Contract

```
typedef struct Contract Contract
```

4.6.2.7 ContractAI

```
typedef enum ContractAI ContractAI
```

4.6.2.8 ContractType

```
typedef enum ContractType ContractType
```

4.6.2.9 Player

```
typedef struct Player Player
```

4.6.2.10 Position

```
typedef enum Position Position
```

4.6.2.11 TextPosition

```
typedef enum TextPosition TextPosition
```

4.6.2.12 Value

```
typedef enum Value Value
```

4.6.3 Enumeration Type Documentation

4.6.3.1 Bool

```
enum Bool
```

Enumerator

FALSE	
TRUE	

4.6.3.2 CardAI

`enum CardAI`

Enumerator

CARD_USER	
CARD_AI_FIRSTAVAILABLE	
CARD_AI_STANDARD	

4.6.3.3 Coinche

`enum Coinche`

Enumerator

NOT_COINCHED	
COINCHED	
OVERCOINCHED	

4.6.3.4 Color

`enum Color`

Enumerator

NULL_COLOR	
SPADE	
HEART	
DIAMOND	
CLUB	
ALLTRUMP	
NOTRUMP	

4.6.3.5 ContractAI

enum ContractAI

Enumerator

CONTRACT_USER	
CONTRACT_AI_ALWAYSEIGHTY	
CONTRACT_AI_STANDARD	

4.6.3.6 ContractType

enum ContractType

Enumerator

POINTS	
CAPOT	
GENERAL	

4.6.3.7 Position

enum Position

Enumerator

SOUTH	
WEST	
NORTH	
EAST	

4.6.3.8 TextPosition

enum TextPosition

Enumerator

TEXT_LEFT	
TEXT_CENTER	
TEXT_RIGHT	

4.6.3.9 Value

```
enum Value
```

Enumerator

NULL_VALUE	
SEVEN	
EIGHT	
NINE	
JACK	
QUEEN	
KING	
TEN	
ACE	

4.6.4 Variable Documentation

4.6.4.1 CARD_POINTS_TABLE

```
const int CARD_POINTS_TABLE[4][8]
```

4.6.4.2 CARDAI_STR_TABLE

```
const char* CARDAI_STR_TABLE[NB_CARD_AI]
```

4.6.4.3 COINCHE_STR_TABLE

```
const char* COINCHE_STR_TABLE[3]
```

4.6.4.4 COLOR_STR_TABLE

```
const char* COLOR_STR_TABLE[7][2]
```

4.6.4.5 CONTRACTAI_STR_TABLE

```
const char* CONTRACTAI_STR_TABLE[NB_CONTRACT_AI]
```

4.6.4.6 CONTRACTTYPE_STR_TABLE

```
const char* CONTRACTTYPE_STR_TABLE[3]
```

4.6.4.7 VALUE_STR_TABLE

```
const char* VALUE_STR_TABLE[9][2]
```

4.7 F:/C_IFE/IFE-coinche/src/displayMain.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "displayMain.h"
#include "leaderboard.h"
#include "stringUtils.h"
```

Include dependency graph for displayMain.c:

Functions

- void displayFrame (void)
- void clearInfoMsg (void)
- void displayInfoMsg (char messageLine1[ ], char messageLine2[ ])
- void resizeCmdWindow (int nbOfLines, int nbOfColumns)
- void displayMenu (void)
- void displayLeaderboard (void)
- void displayCredits (void)

4.7.1 Function Documentation

4.7.1.1 clearInfoMsg()

```
void clearInfoMsg (
            void  )
```

Clears the info box. The cursor is set to the middle of the first line Here is the caller graph for this function:

4.7.1.2 displayCredits()

```
void displayCredits (
            void  )
```

Displays the credits of this project Here is the call graph for this function: Here is the caller graph for this function:

4.7.1.3 displayFrame()

```
void displayFrame (
            void  )
```

Displays the empty frame needed for every other display (and replaces whatever was there before) Here is the caller graph for this function:

4.7.1.4 displayInfoMsg()

```
void displayInfoMsg (
            char messageLine1[],
            char messageLine2[] )
```

Parameters

messageLine2[]	the second line of the message to display
messageLine1[]	the first line of the message to display

Displays a centered message in the info box. The cursor is left to the end of the message Here is the call graph for this function: Here is the caller graph for this function:

4.7.1.5 displayLeaderboard()

```
void displayLeaderboard (
            void  )
```

Displays the leaderboard of the top ten in number of wins Here is the call graph for this function: Here is the caller graph for this function:

4.7.1.6 displayMenu()

```
void displayMenu (
            void  )
```

Displays the menu Here is the call graph for this function: Here is the caller graph for this function:

4.7.1.7 resizeCmdWindow()

```
void resizeCmdWindow (
            int nbOfLines,
            int nbOfColumns )
```

Parameters

nbOfLines	the number of lines that should be displayed
nbOfColumns	the number of lines that should be displayed. Microsoft docs recommands a value between 40 and 135

Resizes the command prompt window to a given number of lines and columns Here is the caller graph for this function:

4.8 F:/C_IFE/IFE-coinche/src/displayMain.h File Reference

`#include "core.h"`
Include dependency graph for displayMain.h: This graph shows which files directly or indirectly include this file:

Functions

- void displayFrame (void)
- void clearInfoMsg (void)
- void displayInfoMsg (char messageLine1[], char messageLine2[])
- void resizeCmdWindow (int nbOfLines, int nbOfColumns)
- void displayMenu (void)
- void displayLeaderboard (void)
- void displayCredits (void)

4.8.1 Function Documentation

4.8.1.1 clearInfoMsg()

```
void clearInfoMsg (
            void  )
```

Clears the info box. The cursor is set to the middle of the first line Here is the caller graph for this function:

4.8.1.2 displayCredits()

```
void displayCredits (
            void  )
```

Displays the credits of this project Here is the call graph for this function: Here is the caller graph for this function:

4.8.1.3 displayFrame()

```
void displayFrame (
            void  )
```

Displays the empty frame needed for every other display (and replaces whatever was there before) Here is the caller graph for this function:

4.8.1.4 displayInfoMsg()

```
void displayInfoMsg (
            char messageLine1[],
            char messageLine2[] )
```

Parameters

messageLine2[ ]	the second line of the message to display
messageLine1[ ]	the first line of the message to display

Displays a centered message in the info box. The cursor is left to the end of the message Here is the call graph for this function: Here is the caller graph for this function:

4.8.1.5 displayLeaderboard()

```
void displayLeaderboard (
            void  )
```

Displays the leaderboard of the top ten in number of wins Here is the call graph for this function: Here is the caller graph for this function:

4.8.1.6 displayMenu()

```
void displayMenu (
            void  )
```

Displays the menu Here is the call graph for this function: Here is the caller graph for this function:

4.8.1.7 resizeCmdWindow()

```
void resizeCmdWindow (
            int nbOfLines,
            int nbOfColumns )
```

Parameters

nbOfLines	the number of lines that should be displayed
nbOfColumns	the number of lines that should be displayed. Microsoft docs recommands a value between 40 and 135

Resizes the command prompt window to a given number of lines and columns Here is the caller graph for this function:

4.9 F:/C_IFE/IFE-coinche/src/displayRound.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include "displayRound.h"
#include "stringUtils.h"
```

Include dependency graph for displayRound.c:

Functions

- void displayEmptyCard (void)
- void deleteCardDisplay (void)
- void clearCardDisplay (void)
- void changeCardDisplay (Card card)
- void preparePlayTable (void)
- void clearTopRightBox (void)
- void prepareLastTrickDisplay (void)
- void displayBiddingMenuLine (char lineToDisplay[ ], Bool firstLine, Bool lastLine)
- void clearContractDisplay (void)
- void updateContractDisplay (char playerName[ ], Contract contract)
- void displayPlayerName (Player player, Bool underline)
- void updateRoundNbDisplay (int roundNb)
- void updateTrickNbDisplay (int trickNb)
- void updateTeamScore (Player players[ ])
- void updateLastTrickDisplay (Card lastTrickCards[ ], Position startingPlayer)
- void clearLastTrickDisplay (void)
- void displayTrickCard (Card playedCard, Position currentPlayer)
- void deleteDisplayedTrickCards (void)
- void displayNumbersAbovePlayerHand (Card cardsInHand[ ], int nbOfCardsInHand)
- void displayPlayerHand (Card cardsInHand[ ], int nbOfCardsInHand)
- void deletePlayerHand (void)
- void updatePlayerTrickPoints (int points, Position playerPos)
- void clearDisplayedTrickPoints (void)

4.9.1 Function Documentation

4.9.1.1 changeCardDisplay()

```
void changeCardDisplay (
            Card card )
```

Parameters

card	the card to display. Only its value and color matter. Depending on its color, the card will be displayed in red or white

Changes a card at the current cursor position (the cursor comes back to that position afterwards) Here is the caller graph for this function:

4.9.1.2 clearCardDisplay()

```
void clearCardDisplay (
            void  )
```

Clears a card at the current cursor position (the cursor comes back to that position afterwards) Here is the caller graph for this function:

4.9.1.3 clearContractDisplay()

```
void clearContractDisplay (
            void  )
```

Clears contract display. The cursor is left untouched Here is the caller graph for this function:

4.9.1.4 clearDisplayedTrickPoints()

```
void clearDisplayedTrickPoints (
            void  )
```

Clears all 4 displayed trick points Here is the caller graph for this function:

4.9.1.5 clearLastTrickDisplay()

```
void clearLastTrickDisplay (
            void  )
```

Clears the 4 cards in the last trick display Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.6 clearTopRightBox()

```
void clearTopRightBox (
            void  )
```

Clears top-right box. The cursor is left untouched Here is the caller graph for this function:

4.9.1.7 deleteCardDisplay()

```
void deleteCardDisplay (
            void  )
```

Deletes a card at the current cursor position (the cursor comes back to that position afterwards) Here is the caller graph for this function:

4.9.1.8 deleteDisplayedTrickCards()

```
void deleteDisplayedTrickCards (
            void  )
```

Deletes all 4 currently displayed trick cards Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.9 deletePlayerHand()

```
void deletePlayerHand (
            void  )
```

Deletes the content of the five lines (including the numbers line) containing the displayed player hand Here is the caller graph for this function:

4.9.1.10 displayBiddingMenuLine()

```
void displayBiddingMenuLine (
            char lineToDisplay[],
            Bool firstLine,
            Bool lastLine )
```

Parameters

lineToDisplay[]	the string to display, must contain a maximum of 15 characters
firstline	set to TRUE if this is the first line of the menu to display, set to FALSE otherwise
lastLine	set to TRUE if this is the last line of the menu to display, set to FALSE otherwise

Displays a single line in the top-right box. The cursor is moved to the next line Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.11 displayEmptyCard()

```
void displayEmptyCard (
            void  )
```

Displays an empty card at the current cursor position (the cursor comes back to that position afterwards) Here is the caller graph for this function:

4.9.1.12 displayNumbersAbovePlayerHand()

```
void displayNumbersAbovePlayerHand (
            Card cardsInHand[],
            int nbOfCardsInHand )
```

Parameters

cardInHand[]	array containing the player's cards (only the canPlay property is looked at)
nbOfCardsInHand	the number of cards in the SOUTH player's hand

Displays a line of sequencial numbers above the player's hand cards that can be played Here is the caller graph for this function:

4.9.1.13 displayPlayerHand()

```
void displayPlayerHand (
            Card cardsInHand[],
            int nbOfCardsInHand )
```

Parameters

cardsInHand[]	array containing the player's cards
nbOfCardsInHand	the number of cards in cardsInHand

Displays the player's hand, centered Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.14 displayPlayerName()

```
void displayPlayerName (
```

```
            Player player,
            Bool underline )
```

Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.15 displayTrickCard()

```
void displayTrickCard (
            Card playedCard,
            Position currentPlayer )
```

Parameters

playedCard	the card to display
currentPlayer	position of the player who just played the last card

Displays a trick card in the middle of the table Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.16 prepareLastTrickDisplay()

```
void prepareLastTrickDisplay (
            void  )
```

Displays an empty "Last trick" template in the top-right box. The cursor is left untouched Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.17 preparePlayTable()

```
void preparePlayTable (
            void  )
```

Prepares the play phase by displaying everything needed Here is the caller graph for this function:

4.9.1.18 updateContractDisplay()

```
void updateContractDisplay (
            char playerName[],
            Contract contract )
```

Parameters

playerName	the contract issuer's name
contract	the contract to display

Fills in the contract table corner with a given contract Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.19 updateLastTrickDisplay()

```
void updateLastTrickDisplay (
            Card lastTrickCards[],
            Position startingPlayer )
```

Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.20 updatePlayerTrickPoints()

```
void updatePlayerTrickPoints (
            int points,
            Position playerPos )
```

Parameters

points	the player's trick points
playerPos	the position the player

Displays the current trick points of a player near its name Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.21 updateRoundNbDisplay()

```
void updateRoundNbDisplay (
            int roundNb )
```

Here is the caller graph for this function:

4.9.1.22 updateTeamScore()

```
void updateTeamScore (
            Player players[] )
```

Here is the call graph for this function: Here is the caller graph for this function:

4.9.1.23 updateTrickNbDisplay()

```
void updateTrickNbDisplay (
            int trickNb )
```

Here is the caller graph for this function:

4.10 F:/C_IFE/IFE-coinche/src/displayRound.h File Reference

```
#include "core.h"
```

Include dependency graph for displayRound.h: This graph shows which files directly or indirectly include this file:

Functions

- void displayEmptyCard (void)
- void deleteCardDisplay (void)
- void clearCardDisplay (void)
- void changeCardDisplay (Card card)
- void preparePlayTable (void)
- void clearTopRightBox (void)
- void prepareLastTrickDisplay (void)
- void displayBiddingMenuLine (char lineToDisplay[ ], Bool firstLine, Bool lastLine)
- void clearContractDisplay (void)
- void updateContractDisplay (char playerName[ ], Contract contract)
- void displayPlayerName (Player player, Bool underline)
- void updateRoundNbDisplay (int roundNb)
- void updateTrickNbDisplay (int trickNb)
- void updateTeamScore (Player players[ ])
- void updateLastTrickDisplay (Card lastTrickCards[ ], Position startingPlayer)
- void clearLastTrickDisplay (void)
- void displayTrickCard (Card playedCard, Position currentPlayer)
- void deleteDisplayedTrickCards (void)
- void displayNumbersAbovePlayerHand (Card cardsInHand[ ], int nbOfCardsInHand)
- void displayPlayerHand (Card cardsInHand[ ], int nbOfCardsInHand)
- void deletePlayerHand (void)
- void updatePlayerTrickPoints (int points, Position playerPos)
- void clearDisplayedTrickPoints (void)

4.10.1 Function Documentation

4.10.1.1 changeCardDisplay()

```
void changeCardDisplay (
            Card card )
```

Parameters

card	the card to display. Only its value and color matter. Depending on its color, the card will be displayed in red or white

Changes a card at the current cursor position (the cursor comes back to that position afterwards) Here is the caller graph for this function:

4.10.1.2 clearCardDisplay()

```
void clearCardDisplay (
            void  )
```

Clears a card at the current cursor position (the cursor comes back to that position afterwards) Here is the caller graph for this function:

4.10.1.3 clearContractDisplay()

```
void clearContractDisplay (
            void  )
```

Clears contract display. The cursor is left untouched Here is the caller graph for this function:

4.10.1.4 clearDisplayedTrickPoints()

```
void clearDisplayedTrickPoints (
            void  )
```

Clears all 4 displayed trick points Here is the caller graph for this function:

4.10.1.5 clearLastTrickDisplay()

```
void clearLastTrickDisplay (
            void  )
```

Clears the 4 cards in the last trick display Here is the call graph for this function: Here is the caller graph for this function:

4.10.1.6 clearTopRightBox()

```
void clearTopRightBox (
            void  )
```

Clears top-right box. The cursor is left untouched Here is the caller graph for this function:

4.10.1.7 deleteCardDisplay()

```
void deleteCardDisplay (
            void  )
```

Deletes a card at the current cursor position (the cursor comes back to that position afterwards) Here is the caller graph for this function:

4.10.1.8 deleteDisplayedTrickCards()

```
void deleteDisplayedTrickCards (
            void  )
```

Deletes all 4 currently displayed trick cards Here is the call graph for this function: Here is the caller graph for this function:

4.10.1.9 deletePlayerHand()

```
void deletePlayerHand (
            void  )
```

Deletes the content of the five lines (including the numbers line) containing the displayed player hand Here is the caller graph for this function:

4.10.1.10 displayBiddingMenuLine()

```
void displayBiddingMenuLine (
            char lineToDisplay[],
            Bool firstLine,
            Bool lastLine )
```

Parameters

lineToDisplay[]	the string to display, must contain a maximum of 15 characters
firstline	set to TRUE if this is the first line of the menu to display, set to FALSE otherwise
lastLine	set to TRUE if this is the last line of the menu to display, set to FALSE otherwise

Displays a single line in the top-right box. The cursor is moved to the next line Here is the call graph for this function: Here is the caller graph for this function:

4.10.1.11 displayEmptyCard()

```
void displayEmptyCard (
            void  )
```

Displays an empty card at the current cursor position (the cursor comes back to that position afterwards) Here is the caller graph for this function:

4.10.1.12 displayNumbersAbovePlayerHand()

```
void displayNumbersAbovePlayerHand (
            Card cardsInHand[],
            int nbOfCardsInHand )
```

Parameters

cardInHand[]	array containing the player's cards (only the canPlay property is looked at)
nbOfCardsInHand	the number of cards in the SOUTH player's hand

Displays a line of sequencial numbers above the player's hand cards that can be played Here is the caller graph for this function:

4.10.1.13 displayPlayerHand()

```
void displayPlayerHand (
            Card cardsInHand[],
            int nbOfCardsInHand )
```

Parameters

cardsInHand[]	array containing the player's cards
nbOfCardsInHand	the number of cards in cardsInHand

Displays the player's hand, centered Here is the call graph for this function: Here is the caller graph for this function:

4.10.1.14 displayPlayerName()

```
void displayPlayerName (
```

```
            Player player,
            Bool underline )
```

Here is the call graph for this function: Here is the caller graph for this function:

4.10.1.15 displayTrickCard()

```
void displayTrickCard (
            Card playedCard,
            Position currentPlayer )
```

Parameters

playedCard	the card to display
currentPlayer	position of the player who just played the last card

Displays a trick card in the middle of the table Here is the call graph for this function: Here is the caller graph for this function:

4.10.1.16 prepareLastTrickDisplay()

```
void prepareLastTrickDisplay (
            void  )
```

Displays an empty "Last trick" template in the top-right box. The cursor is left untouched Here is the call graph for this function: Here is the caller graph for this function:

4.10.1.17 preparePlayTable()

```
void preparePlayTable (
            void  )
```

Prepares the play phase by displaying everything needed Here is the caller graph for this function:

4.10.1.18 updateContractDisplay()

```
void updateContractDisplay (
            char playerName[],
            Contract contract )
```

Parameters

playerName	the contract issuer's name
contract	the contract to display

Fills in the contract table corner with a given contract Here is the call graph for this function: Here is the caller graph for this function:

4.10.1.19 updateLastTrickDisplay()

```
void updateLastTrickDisplay (
            Card lastTrickCards[],
            Position startingPlayer )
```

Here is the call graph for this function: Here is the caller graph for this function:

4.10.1.20 updatePlayerTrickPoints()

```
void updatePlayerTrickPoints (
            int points,
            Position playerPos )
```

Parameters

points	the player's trick points
playerPos	the position the player

Displays the current trick points of a player near its name Here is the call graph for this function: Here is the caller graph for this function:

4.10.1.21 updateRoundNbDisplay()

```
void updateRoundNbDisplay (
            int roundNb )
```

Here is the caller graph for this function:

4.10.1.22 updateTeamScore()

```
void updateTeamScore (
            Player players[] )
```

Here is the call graph for this function: Here is the caller graph for this function:

4.10.1.23 updateTrickNbDisplay()

```
void updateTrickNbDisplay (
            int trickNb )
```

Here is the caller graph for this function:

4.11 F:/C_IFE/IFE-coinche/src/leaderboard.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "leaderboard.h"
```

Include dependency graph for leaderboard.c:

Functions

- void writeLine (FILE *leaderboard, char playerName[ ], int wins)
- void increaseWins (char playerName[ ])
- int getTopTen (char names[ ][MAX_PLAYER_NAME_LENGTH+1], int wins[ ])

4.11.1 Function Documentation

4.11.1.1 getTopTen()

```
int getTopTen (
            char names[ ][MAX_PLAYER_NAME_LENGTH+1],
            int wins[ ] )
```

Parameters

names[ ][MAX_PLAYER_NAME_LENGTH+1]	empty two-dimensional array of chars, will be filled with the top ten player names
wins[ ]	empty array of integers, will be filled with the top ten player wins

Returns

nbOfPlayers: the number of players found in the file, from 0 to 10

Retrieves the top ten players in the leaderboard file Here is the caller graph for this function:

4.11.1.2 increaseWins()

```
void increaseWins (
            char playerName[ ] )
```

Parameters

playerName[ ]	name of the player whose number of wins should be increased

Seeks a player to increment its number of wins, and sort the file. If the player isn't in the file, append it at the end Here is the call graph for this function: Here is the caller graph for this function:

4.11.1.3 writeLine()

```
void writeLine (
            FILE * leaderboard,
            char playerName[ ],
            int wins )
```

Parameters

∗leaderboard	leaderboard file pointer
playerName[ ]	name of the player
wins	the player's number of wins

Writes a line at the current cursor position terminated by
Here is the caller graph for this function:

4.12 F:/C_IFE/IFE-coinche/src/leaderboard.h File Reference

`#include "core.h"`
Include dependency graph for leaderboard.h: This graph shows which files directly or indirectly include this file:

Functions

- void writeLine (FILE ∗leaderboard, char playerName[ ], int wins)
- void increaseWins (char playerName[ ])
- int getTopTen (char names[ ][MAX_PLAYER_NAME_LENGTH+1], int wins[ ])

4.12.1 Function Documentation

4.12.1.1 getTopTen()

```
int getTopTen (
            char names[ ][MAX_PLAYER_NAME_LENGTH+1],
            int wins[ ] )
```

Parameters

names[ ][MAX_PLAYER_NAME_LENGTH+1]	empty two-dimensional array of chars, will be filled with the top ten player names
wins[ ]	empty array of integers, will be filled with the top ten player wins

Returns

nbOfPlayers: the number of players found in the file, from 0 to 10

Retrieves the top ten players in the leaderboard file Here is the caller graph for this function:

4.12.1.2 increaseWins()

```
void increaseWins (
            char playerName[ ] )
```

Parameters

playerName[ ]	name of the player whose number of wins should be increased

Seeks a player to increment its number of wins, and sort the file. If the player isn't in the file, append it at the end
Here is the call graph for this function: Here is the caller graph for this function:

4.12.1.3 writeLine()

```
void writeLine (
            FILE * leaderboard,
            char playerName[],
            int wins )
```

Parameters

∗*leaderboard*	leaderboard file pointer
playerName[ ]	name of the player
wins	the player's number of wins

Writes a line at the current cursor position terminated by
Here is the caller graph for this function:

4.13 F:/C_IFE/IFE-coinche/src/main.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "main.h"
#include "displayMain.h"
#include "leaderboard.h"
#include "play.h"
#include "stringUtils.h"
#include "userInput.h"
```

Include dependency graph for main.c:

Functions

- int main (int argc, char ∗argv[ ])
- void mainMenu (Player players[ ])
- void changePlayerSettings (Player players[ ], char quitMsg[ ])
- void setUp (Player players[ ])
- void tearDown (Player players[ ])

4.13.1 Function Documentation

4.13.1.1 changePlayerSettings()

```
void changePlayerSettings (
            Player players[],
            char quitMsg[] )
```

Parameters

players[]	array of 4 players
quitMsg[]	the text displayed as the quit option

Prompts the user to change player settings Here is the call graph for this function: Here is the caller graph for this function:

4.13.1.2 main()

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

Here is the call graph for this function:

4.13.1.3 mainMenu()

```
void mainMenu (
            Player players[] )
```

Parameters

players[]	the four players that will play a game if the user selected 1. Start a game

Executes the action selected by the user in the menu Here is the call graph for this function: Here is the caller graph for this function:

4.13.1.4 setUp()

```
void setUp (
            Player players[] )
```

Parameters

players[]	array of 4 players. Each of them will be initialized

Sets up everything needed for the program to run, including the players Here is the call graph for this function: Here is the caller graph for this function:

4.13.1.5 tearDown()

```
void tearDown (
            Player players[] )
```

Parameters

players[]	array of 4 players

Prepares the program for exit. Mostly frees memory Here is the caller graph for this function:

4.14 F:/C_IFE/IFE-coinche/src/main.h File Reference

```
#include "core.h"
```

Include dependency graph for main.h: This graph shows which files directly or indirectly include this file:

Functions

- void mainMenu (Player players[ ])
- void changePlayerSettings (Player players[ ], char quitMsg[ ])
- void setUp (Player players[ ])
- void tearDown (Player players[ ])

4.14.1 Function Documentation

4.14.1.1 changePlayerSettings()

```
void changePlayerSettings (
            Player players[],
            char quitMsg[] )
```

Parameters

players[]	array of 4 players
quitMsg[]	the text displayed as the quit option

Prompts the user to change player settings Here is the call graph for this function: Here is the caller graph for this function:

4.14.1.2 mainMenu()

```
void mainMenu (
            Player players[] )
```

Parameters

players[]	the four players that will play a game if the user selected 1. Start a game

Executes the action selected by the user in the menu Here is the call graph for this function: Here is the caller graph for this function:

4.14.1.3 setUp()

```
void setUp (
            Player players[] )
```

Parameters

players[]	array of 4 players. Each of them will be initialized

Sets up everything needed for the program to run, including the players Here is the call graph for this function: Here is the caller graph for this function:

4.14.1.4 tearDown()

```
void tearDown (
            Player players[] )
```

Parameters

players[]	array of 4 players

Prepares the program for exit. Mostly frees memory Here is the caller graph for this function:

4.15 F:/C_IFE/IFE-coinche/src/play.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include "play.h"
#include "cardUtils.h"
#include "displayRound.h"
#include "playerUtils.h"
#include "userInput.h"
```

Include dependency graph for play.c:

Functions

- Bool bidAttempt (Player players[ ], Position startingPlayer, Contract ∗contract)
- Contract bidUntilContract (Player players[ ], Position startingPlayer)
- Position playTrick (Player players[ ], Position startingPlayer, Color trump)
- void playRound (Player players[ ], Position startingPlayer, Color trump)
- void awardTeamPoints (Player players[ ], Contract contract)
- int playGame (Player players[ ])
- float playAIGames (Player players[ ], int nbOfGames, int nbOfGamesWon[ ])

4.15.1 Function Documentation

4.15.1.1 awardTeamPoints()

```
void awardTeamPoints (
            Player players[],
            Contract contract )
```

Parameters

players[]	array of 4 players
contract	the contract that was previously made

Do the aftermaths of a round: check whether or not the contract issuer's team fulfilled the contract, and award points accordingly Here is the call graph for this function: Here is the caller graph for this function:

4.15.1.2 bidAttempt()

```
Bool bidAttempt (
            Player players[],
            Position startingPlayer,
            Contract * contract )
```

Parameters

players[]	array of 4 players
startingPlayer	position of the starting player
∗contract	pointer to the contract being debated. Will be edited someone decides to make a contract

Returns

everyonePassed: FALSE if a contract was made, TRUE otherwise

Play a bid attempt: either a contract is made or everyone passed Here is the call graph for this function: Here is the caller graph for this function:

4.15.1.3 bidUntilContract()

```
Contract bidUntilContract (
            Player players[],
            Position startingPlayer )
```

Parameters

players[]	array of 4 players
startingPlayer	position of the starting player

Returns

contract: the contract that was made

Deal cards and play bid attempts until a contract is made Here is the call graph for this function: Here is the caller graph for this function:

4.15.1.4 playAIGames()

```
float playAIGames (
            Player players[],
            int nbOfGames,
            int nbOfGamesWon[] )
```

Parameters

players[]	array of 4 players
nbOfGames	the number of games to be played
nbOfGamesWon[]	empty array of length 2. The number of game won by each team will be written in it

Returns

averageGameLength: average number of rounds played for each game

Plays a given number of AI games and returns some stats Here is the call graph for this function: Here is the caller graph for this function:

4.15.1.5 playGame()

```
int playGame (
            Player players[] )
```

Parameters

players[]	array of 4 players

Returns

currentRound: the number of rounds played

Plays a full game until a team wins (it reaches 701 points) Here is the call graph for this function: Here is the caller graph for this function:

4.15.1.6 playRound()

```
void playRound (
            Player players[],
            Position startingPlayer,
            Color trump )
```

Parameters

players[]	array of 4 players
startingPlayer	position of the starting player
trump	the current trump

Plays a full 8-trick round, counting points Here is the call graph for this function: Here is the caller graph for this function:

4.15.1.7 playTrick()

```
Position playTrick (
            Player players[],
            Position startingPlayer,
            Color trump )
```

Parameters

players[]	array of 4 players
startingPlayer	position of the starting player
trump	the current trump

Returns

trickWinner: position of the winner of the trick

Plays a single trick : each player plays a card, then the player with the strongest card wins the trick and gets points Here is the call graph for this function: Here is the caller graph for this function:

4.16 F:/C_IFE/IFE-coinche/src/play.h File Reference

`#include "core.h"`
Include dependency graph for play.h: This graph shows which files directly or indirectly include this file:

Functions

- Bool bidAttempt (Player players[ ], Position startingPlayer, Contract ∗contract)
- Contract bidUntilContract (Player players[ ], Position startingPlayer)
- Position playTrick (Player players[ ], Position startingPlayer, Color trump)
- void playRound (Player players[ ], Position startingPlayer, Color trump)
- void awardTeamPoints (Player players[ ], Contract contract)
- int playGame (Player players[ ])
- float playAIGames (Player players[ ], int nbOfGames, int nbOfGamesWon[ ])

4.16.1 Function Documentation

4.16.1.1 awardTeamPoints()

```
void awardTeamPoints (
            Player players[],
            Contract contract )
```

Parameters

players[]	array of 4 players
contract	the contract that was previously made

Do the aftermaths of a round: check whether or not the contract issuer's team fulfilled the contract, and award points accordingly Here is the call graph for this function: Here is the caller graph for this function:

4.16.1.2 bidAttempt()

```
Bool bidAttempt (
            Player players[],
            Position startingPlayer,
            Contract * contract )
```

Parameters

players[]	array of 4 players
startingPlayer	position of the starting player
∗contract	pointer to the contract being debated. Will be edited someone decides to make a contract

Returns

everyonePassed: FALSE if a contract was made, TRUE otherwise

Play a bid attempt: either a contract is made or everyone passed Here is the call graph for this function: Here is the caller graph for this function:

4.16.1.3 bidUntilContract()

```
Contract bidUntilContract (
            Player players[],
            Position startingPlayer )
```

Parameters

players[]	array of 4 players
startingPlayer	position of the starting player

Returns

contract: the contract that was made

Deal cards and play bid attempts until a contract is made Here is the call graph for this function: Here is the caller graph for this function:

4.16.1.4 playAIGames()

```
float playAIGames (
            Player players[],
            int nbOfGames,
            int nbOfGamesWon[] )
```

Parameters

players[]	array of 4 players
nbOfGames	the number of games to be played
nbOfGamesWon[]	empty array of length 2. The number of game won by each team will be written in it

Returns

averageGameLength: average number of rounds played for each game

Plays a given number of AI games and returns some stats Here is the call graph for this function: Here is the caller graph for this function:

4.16.1.5 playGame()

```
int playGame (
            Player players[] )
```

Parameters

players[]	array of 4 players

Returns

currentRound: the number of rounds played

Plays a full game until a team wins (it reaches 701 points) Here is the call graph for this function: Here is the caller graph for this function:

4.16.1.6 playRound()

```
void playRound (
            Player players[],
            Position startingPlayer,
            Color trump )
```

Parameters

players[]	array of 4 players
startingPlayer	position of the starting player
trump	the current trump

Plays a full 8-trick round, counting points Here is the call graph for this function: Here is the caller graph for this function:

4.16.1.7 playTrick()

```
Position playTrick (
            Player players[],
            Position startingPlayer,
            Color trump )
```

Parameters

players[]	array of 4 players
startingPlayer	position of the starting player
trump	the current trump

Returns

trickWinner: position of the winner of the trick

Plays a single trick : each player plays a card, then the player with the strongest card wins the trick and gets points Here is the call graph for this function: Here is the caller graph for this function:

4.17 F:/C_IFE/IFE-coinche/src/playerUtils.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "playerUtils.h"
#include "ai.h"
#include "cardUtils.h"
#include "userInput.h"
```

Include dependency graph for playerUtils.c:

Functions

- Card getPlayerCard (Player *player, Card trickCards[ ], int nbOfTrickCards, Color trump, Color roundColor)
- Bool getPlayerContract (Player player, Contract *contract)
- void cardsDistribution (Player players[ ])
- int getTeamRoundPoints (Player players[ ], Position player)
- void increaseTeamTotalScore (Player players[ ], Position player, int roundScore)

4.17.1 Function Documentation

4.17.1.1 cardsDistribution()

```
void cardsDistribution (
            Player players[] )
```

Parameters

players[]	array of 4 players

Function that randomly deals 8 of the 32 cards to each player at the beginning of bidding Here is the call graph for this function: Here is the caller graph for this function:

4.17.1.2 getPlayerCard()

```
Card getPlayerCard (
            Player * player,
            Card trickCards[],
            int nbOfTrickCards,
            Color trump,
            Color roundColor )
```

Parameters

∗player	pointer to the player who has to choose a card. Note: the "canPlay" flag for the player cards has to be set already
trickCards[]	the cards already played by the previous players. Only the first N cards matter, where N is nbOfTrickCards
nbOfTrickCards	the number of cards played by the previous players. Can be 0
trump	the current trump
roundColor	the color of the first played card in the round

Returns

chosenCard: the chosen card

Get the player to choose a card among the ones that can be played, depending on the player type (User/AI) Here is the call graph for this function: Here is the caller graph for this function:

4.17.1.3 getPlayerContract()

```
Bool getPlayerContract (
            Player player,
            Contract * contract )
```

Parameters

player	the player who has to make a decision
∗contract	pointer to the contract being debated. Will be edited if the player decides to make a contract

Returns

hasPassed: TRUE if the player has decided to pass, FALSE if the player decided to make a contract

Get the player to decide whether or not to make a contract, depending on the player type (User/AI) Here is the call graph for this function: Here is the caller graph for this function:

4.17.1.4 getTeamRoundPoints()

```
int getTeamRoundPoints (
            Player players[],
            Position player )
```

Parameters

players[]	array of 4 players
player	position of a player in the team of interest

Returns

roundPoints: the total round points of the team

Get a team's total trick points from the previous round Here is the caller graph for this function:

4.17.1.5 increaseTeamTotalScore()

```
void increaseTeamTotalScore (
            Player players[],
            Position player,
            int roundScore )
```

Parameters

players[]	array of 4 players
player	position of a player in the team of interest
roundScore	how much the team's total score should be increased by

Increase a team's total score Here is the caller graph for this function:

4.18 F:/C_IFE/IFE-coinche/src/playerUtils.h File Reference

```
#include "core.h"
```

Include dependency graph for playerUtils.h: This graph shows which files directly or indirectly include this file:

Functions

- Card getPlayerCard (Player ∗player, Card trickCards[ ], int nbOfTrickCards, Color trump, Color roundColor)
- Bool getPlayerContract (Player player, Contract ∗contract)
- void cardsDistribution (Player players[ ])
- int getTeamRoundPoints (Player players[ ], Position player)
- void increaseTeamTotalScore (Player players[ ], Position player, int roundScore)

4.18.1 Function Documentation

4.18.1.1 cardsDistribution()

```
void cardsDistribution (
            Player players[ ] )
```

Parameters

players[ ]	array of 4 players

Function that randomly deals 8 of the 32 cards to each player at the beginning of bidding Here is the call graph for this function: Here is the caller graph for this function:

4.18.1.2 getPlayerCard()

```
Card getPlayerCard (
            Player * player,
            Card trickCards[],
            int nbOfTrickCards,
            Color trump,
            Color roundColor )
```

Parameters

∗*player*	pointer to the player who has to choose a card. Note: the "canPlay" flag for the player cards has to be set already
trickCards[ ]	the cards already played by the previous players. Only the first N cards matter, where N is nbOfTrickCards
nbOfTrickCards	the number of cards played by the previous players. Can be 0
trump	the current trump
roundColor	the color of the first played card in the round

Returns

chosenCard: the chosen card

Get the player to choose a card among the ones that can be played, depending on the player type (User/AI) Here is the call graph for this function: Here is the caller graph for this function:

4.18.1.3 getPlayerContract()

```
Bool getPlayerContract (
            Player player,
            Contract * contract )
```

Parameters

player	the player who has to make a decision
∗contract	pointer to the contract being debated. Will be edited if the player decides to make a contract

Returns

hasPassed: TRUE if the player has decided to pass, FALSE if the player decided to make a contract

Get the player to decide whether or not to make a contract, depending on the player type (User/AI) Here is the call graph for this function: Here is the caller graph for this function:

4.18.1.4 getTeamRoundPoints()

```
int getTeamRoundPoints (
            Player players[],
            Position player )
```

Parameters

players[]	array of 4 players
player	position of a player in the team of interest

Returns

roundPoints: the total round points of the team

Get a team's total trick points from the previous round Here is the caller graph for this function:

4.18.1.5 increaseTeamTotalScore()

```
void increaseTeamTotalScore (
            Player players[],
            Position player,
            int roundScore )
```

Parameters

players[]	array of 4 players
player	position of a player in the team of interest
roundScore	how much the team's total score should be increased by

Increase a team's total score Here is the caller graph for this function:

4.19 F:/C_IFE/IFE-coinche/src/stringUtils.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include "stringUtils.h"
```

Include dependency graph for stringUtils.c:

Macros

- #define UNDERLINE_SEQUENCE_LENGTH 9

Functions

- char * cropStr (const char string[], int maxLength)
- char * formatStr (char string[], int maxLength, TextPosition textPosition, Bool underline)

4.19.1 Macro Definition Documentation

4.19.1.1 UNDERLINE_SEQUENCE_LENGTH

```
#define UNDERLINE_SEQUENCE_LENGTH 9
```

4.19.2 Function Documentation

4.19.2.1 cropStr()

```
char * cropStr (
            const char string[],
            int maxLength )
```

Parameters

string[]	the string to crop
maxLength	the maximum length the string can be without cropping

Returns

croppedString∗: pointer to the first char of the cropped string. Must be freed eventually!

Takes a string and returns a cropped version of it with dots at the end if it exceeds a given length Here is the caller graph for this function:

4.19.2.2 formatStr()

```
formatStr (
            char string[],
            int maxLength,
            TextPosition textPosition,
            Bool underline )
```

Parameters

string[]	the string to center
maxLength	the maximum length of the space the centered string will be displayed in
textPosition	text alignement within the string (left, center, right)
underline	TRUE if the text sould be underlined, FALSE otherwise

Returns

formattedString∗: pointer to the first char of the centered string. Must be freed eventually!

Takes a string and returns a fixed length, text-aligned version of it, cropped and or underlined if needed Here is the call graph for this function: Here is the caller graph for this function:

4.20 F:/C_IFE/IFE-coinche/src/stringUtils.h File Reference

```
#include "core.h"
```

Include dependency graph for stringUtils.h: This graph shows which files directly or indirectly include this file:

Functions

- char ∗ cropStr (const char string[ ], int maxLength)
- char ∗ formatStr (char string[ ], int maxLength, TextPosition textPosition, Bool underline)

4.20.1 Function Documentation

4.20.1.1 cropStr()

```
char* cropStr (
            const char string[],
            int maxLength )
```

Parameters

string[]	the string to crop
maxLength	the maximum length the string can be without cropping

Returns

croppedString*: pointer to the first char of the cropped string. Must be freed eventually!

Takes a string and returns a cropped version of it with dots at the end if it exceeds a given length Here is the caller graph for this function:

4.20.1.2 formatStr()

```
char* formatStr (
            char string[],
            int maxLength,
            TextPosition textPosition,
            Bool underline )
```

Parameters

string[]	the string to center
maxLength	the maximum length of the space the centered string will be displayed in
textPosition	text alignement within the string (left, center, right)
underline	TRUE if the text sould be underlined, FALSE otherwise

Returns

formattedString*: pointer to the first char of the centered string. Must be freed eventually!

Takes a string and returns a fixed length, text-aligned version of it, cropped and or underlined if needed Here is the call graph for this function: Here is the caller graph for this function:

4.21 F:/C_IFE/IFE-coinche/src/userInput.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "userInput.h"
#include "displayMain.h"
#include "displayRound.h"
```

Include dependency graph for userInput.c:

Functions

- char * inputUserStr (int maxStrLength, char displayStrline1[ ], char displayStrline2[ ], Bool useSecondLine↩ AsInput)
- int inputUserInt (int minBound, int maxBound, char displayStr[ ])
- void inputUserAcknowledgement (char displayMsg[ ])
- char * inputUserName (char displayMsg[ ])
- Card askUserCard (Card cardArray[ ], int nbOfCards)
- Bool askUserContract (Contract *contract)

4.21.1 Function Documentation

4.21.1.1 askUserCard()

```
Card askUserCard (
            Card cardArray[ ],
            int nbOfCards )
```

Parameters

cardArray[ ]	array containing the player's cards
nbOfCards	the number of cards in cardArray

Asks the user for a card to play among the valid ones, according to the Coinche rules Here is the call graph for this function: Here is the caller graph for this function:

4.21.1.2 askUserContract()

```
Bool askUserContract (
            Contract * contract )
```

Parameters

**contract*	pointer to the contract being debated

Asks the user for their action during their bidding turn, according to the Coinche rules Here is the call graph for this function: Here is the caller graph for this function:

4.21.1.3 inputUserAcknowledgement()

```
void inputUserAcknowledgement (
            char displayMsg[ ] )
```

Parameters

displayMsg[ ]	the optional message to display. To ignore this argument, set it to ""

Asks the user to press enter, and eventually display a message along with it Here is the call graph for this function: Here is the caller graph for this function:

4.21.1.4 inputUserInt()

```
int inputUserInt (
            int minBound,
            int maxBound,
            char displayStr[] )
```

Parameters

minBound	the minimum valid value the user can enter
maxBound	the maximum valid value the user can enter
displayStr[]	the prompt given to the user asking for an input

Returns

userVal: the value entered by the user, guaranteed to be an int between minBound and maxBound

Asks the user for an int between two bounds Here is the call graph for this function: Here is the caller graph for this function:

4.21.1.5 inputUserName()

```
char * inputUserName (
            char displayMsg[] )
```

Parameters

displayMsg[]	the prompt given to the user asking for a name

Returns

∗userName: pointer to the first char of the user name. Must be freed eventually!

Asks the user to enter a player name. Prevent the user from entering ';' or any character not in the ASCII range 32-126 Here is the call graph for this function: Here is the caller graph for this function:

4.21.1.6 inputUserStr()

```
char * inputUserStr (
            int maxStrLength,
            char displayStrline1[],
            char displayStrline2[],
            Bool useSecondLineAsInput )
```

Parameters

maxStrLength	the maximum length of the input string, including the terminating \0. Anything bigger than that will be cropped
displayStrline1[ ]	first line of the prompt given to the user asking for an input
displayStrline2[ ]	second line of the prompt given to the user. If useSecondLineAsInput is set to true, this is ignored
useSecondLineAsInput	if set to TRUE, the user will input the string on an empty second line instead of the end of the current line

Returns

∗inputStr: pointer to the first char of the user string. Must be freed eventually!

Asks the user to input a string Here is the call graph for this function: Here is the caller graph for this function:

4.22 F:/C_IFE/IFE-coinche/src/userInput.h File Reference

`#include "core.h"`
Include dependency graph for userInput.h: This graph shows which files directly or indirectly include this file:

Functions

- char ∗ inputUserStr (int maxStrLength, char displayStrline1[ ], char displayStrline2[ ], Bool useSecondLine↩ AsInput)
- int inputUserInt (int minBound, int maxBound, char displayStr[ ])
- void inputUserAcknowledgement (char displayMsg[ ])
- char ∗ inputUserName (char displayMsg[ ])
- Card askUserCard (Card cardArray[ ], int nbOfCards)
- Bool askUserContract (Contract ∗contract)

4.22.1 Function Documentation

4.22.1.1 askUserCard()

```
Card askUserCard (
            Card cardArray[],
            int nbOfCards )
```

Parameters

cardArray[ ]	array containing the player's cards
nbOfCards	the number of cards in cardArray

Asks the user for a card to play among the valid ones, according to the Coinche rules Here is the call graph for this function: Here is the caller graph for this function:

4.22.1.2 askUserContract()

```
Bool askUserContract (
            Contract * contract )
```

Parameters

∗contract	pointer to the contract being debated

Asks the user for their action during their bidding turn, according to the Coinche rules Here is the call graph for this function: Here is the caller graph for this function:

4.22.1.3 inputUserAcknowledgement()

```
void inputUserAcknowledgement (
            char displayMsg[] )
```

Parameters

displayMsg[]	the optional message to display. To ignore this argument, set it to ""

Asks the user to press enter, and eventually display a message along with it Here is the call graph for this function: Here is the caller graph for this function:

4.22.1.4 inputUserInt()

```
int inputUserInt (
            int minBound,
            int maxBound,
            char displayStr[] )
```

Parameters

minBound	the minimum valid value the user can enter
maxBound	the maximum valid value the user can enter
displayStr[]	the prompt given to the user asking for an input

Returns

userVal: the value entered by the user, guaranteed to be an int between minBound and maxBound

Asks the user for an int between two bounds Here is the call graph for this function: Here is the caller graph for this function:

4.22.1.5 inputUserName()

```
char* inputUserName (
            char displayMsg[] )
```

Parameters

displayMsg[]	the prompt given to the user asking for a name

Returns

∗userName: pointer to the first char of the user name. Must be freed eventually!

Asks the user to enter a player name. Prevent the user from entering ';' or any character not in the ASCII range 32-126 Here is the call graph for this function: Here is the caller graph for this function:

4.22.1.6 inputUserStr()

```
char* inputUserStr (
            int maxStrLength,
            char displayStrline1[],
            char displayStrline2[],
            Bool useSecondLineAsInput )
```

Parameters

maxStrLength	the maximum length of the input string, including the terminating \0. Anything bigger than that will be cropped
displayStrline1[]	first line of the prompt given to the user asking for an input
displayStrline2[]	second line of the prompt given to the user. If useSecondLineAsInput is set to true, this is ignored
useSecondLineAsInput	if set to TRUE, the user will input the string on an empty second line instead of the end of the current line

Returns

∗inputStr: pointer to the first char of the user string. Must be freed eventually!

Asks the user to input a string Here is the call graph for this function: Here is the caller graph for this function:

Index