

Coinche Project Documentation

Generated by Doxygen 1.8.18

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 Card Struct Reference	5
3.1.1 Field Documentation	5
3.1.1.1 canPlay	5
3.1.1.2 color	5
3.1.1.3 value	5
3.2 Contract Struct Reference	6
3.2.1 Field Documentation	6
3.2.1.1 coinche	6
3.2.1.2 issuer	6
3.2.1.3 points	6
3.2.1.4 trump	6
3.2.1.5 type	6
3.3 Player Struct Reference	7
3.3.1 Field Documentation	7
3.3.1.1 cardAI	7
3.3.1.2 cards	7
3.3.1.3 contractAI	7
3.3.1.4 croppedName	7
3.3.1.5 name	8
3.3.1.6 nbOfCards	8
3.3.1.7 pos	8
3.3.1.8 score	8
3.3.1.9 teamScore	8
4 File Documentation	9
4.1 F:/C_IFE/IFE-coinche/src/ai.c File Reference	9
4.2 F:/C_IFE/IFE-coinche/src/ai.h File Reference	9
4.2.1 Function Documentation	9
4.2.1.1 getAICardFirstAvailable()	9
4.2.1.2 getAICardStandard()	10
4.2.1.3 getAIContractAlwaysEighty()	10
4.2.1.4 getAIContractStandard()	11
4.3 F:/C_IFE/IFE-coinche/src/cardUtils.c File Reference	11
4.3.1 Function Documentation	12
4.3.1.1 createDeck()	12
4.3.1.2 findValidCardsInHand()	12

4.3.1.3	getCardArrayPoints()	12
4.3.1.4	getCardPoints()	13
4.3.1.5	getCardStrength()	13
4.3.1.6	getPlayableCards()	13
4.3.1.7	getStrongestCard()	14
4.3.1.8	removeCard()	14
4.3.1.9	setCanPlay()	15
4.3.1.10	sortCards()	15
4.4	F:/C_IFE/IFE-coinche/src/cardUtils.h File Reference	16
4.4.1	Function Documentation	16
4.4.1.1	createDeck()	16
4.4.1.2	findValidCardsInHand()	16
4.4.1.3	getCardArrayPoints()	17
4.4.1.4	getCardPoints()	17
4.4.1.5	getCardStrength()	18
4.4.1.6	getPlayableCards()	18
4.4.1.7	getStrongestCard()	18
4.4.1.8	removeCard()	19
4.4.1.9	setCanPlay()	19
4.4.1.10	sortCards()	20
4.5	F:/C_IFE/IFE-coinche/src/core.c File Reference	20
4.5.1	Variable Documentation	21
4.5.1.1	CARD_POINTS_TABLE	21
4.5.1.2	CARDAI_STR_TABLE	21
4.5.1.3	COINCHE_STR_TABLE	21
4.5.1.4	COLOR_STR_TABLE	21
4.5.1.5	CONTRACTAI_STR_TABLE	22
4.5.1.6	CONTRACTTYPE_STR_TABLE	22
4.5.1.7	VALUE_STR_TABLE	22
4.6	F:/C_IFE/IFE-coinche/src/core.h File Reference	22
4.6.1	Macro Definition Documentation	23
4.6.1.1	MAX_PLAYER_NAME_LENGTH	23
4.6.1.2	NB_CARD_AI	24
4.6.1.3	NB_CONTRACT_AI	24
4.6.2	Typedef Documentation	24
4.6.2.1	Bool	24
4.6.2.2	Card	24
4.6.2.3	CardAI	24
4.6.2.4	Coinche	24
4.6.2.5	Color	24
4.6.2.6	Contract	25
4.6.2.7	ContractAI	25

4.6.2.8 ContractType	25
4.6.2.9 Player	25
4.6.2.10 Position	25
4.6.2.11 TextPosition	25
4.6.2.12 Value	25
4.6.3 Enumeration Type Documentation	25
4.6.3.1 Bool	25
4.6.3.2 CardAI	26
4.6.3.3 Coinche	26
4.6.3.4 Color	26
4.6.3.5 ContractAI	27
4.6.3.6 ContractType	27
4.6.3.7 Position	27
4.6.3.8 TextPosition	27
4.6.3.9 Value	28
4.6.4 Variable Documentation	28
4.6.4.1 CARD_POINTS_TABLE	28
4.6.4.2 CARDAI_STR_TABLE	28
4.6.4.3 COINCHE_STR_TABLE	28
4.6.4.4 COLOR_STR_TABLE	28
4.6.4.5 CONTRACTAI_STR_TABLE	29
4.6.4.6 CONTRACTTYPE_STR_TABLE	29
4.6.4.7 VALUE_STR_TABLE	29
4.7 F:/C_IFE/IFE-coinche/src/displayMain.c File Reference	29
4.7.1 Function Documentation	29
4.7.1.1 clearInfoMsg()	29
4.7.1.2 displayCredits()	30
4.7.1.3 displayFrame()	30
4.7.1.4 displayInfoMsg()	30
4.7.1.5 displayLeaderboard()	30
4.7.1.6 displayMenu()	30
4.7.1.7 resizeCmdWindow()	30
4.8 F:/C_IFE/IFE-coinche/src/displayMain.h File Reference	31
4.8.1 Function Documentation	31
4.8.1.1 clearInfoMsg()	31
4.8.1.2 displayCredits()	31
4.8.1.3 displayFrame()	31
4.8.1.4 displayInfoMsg()	31
4.8.1.5 displayLeaderboard()	32
4.8.1.6 displayMenu()	32
4.8.1.7 resizeCmdWindow()	32
4.9 F:/C_IFE/IFE-coinche/src/displayRound.c File Reference	32

4.9.1 Function Documentation	33
4.9.1.1 changeCardDisplay()	33
4.9.1.2 clearCardDisplay()	33
4.9.1.3 clearContractDisplay()	34
4.9.1.4 clearDisplayedTrickPoints()	34
4.9.1.5 clearLastTrickDisplay()	34
4.9.1.6 clearTopRightBox()	34
4.9.1.7 deleteCardDisplay()	34
4.9.1.8 deleteDisplayedTrickCards()	34
4.9.1.9 deletePlayerHand()	34
4.9.1.10 displayBiddingMenuLine()	34
4.9.1.11 displayEmptyCard()	35
4.9.1.12 displayNumbersAbovePlayerHand()	35
4.9.1.13 displayPlayerHand()	35
4.9.1.14 displayPlayerName()	35
4.9.1.15 displayTrickCard()	36
4.9.1.16 prepareLastTrickDisplay()	36
4.9.1.17 preparePlayTable()	36
4.9.1.18 updateContractDisplay()	36
4.9.1.19 updateLastTrickDisplay()	37
4.9.1.20 updatePlayerTrickPoints()	37
4.9.1.21 updateRoundNbDisplay()	37
4.9.1.22 updateTeamScore()	37
4.9.1.23 updateTrickNbDisplay()	37
4.10 F:/C_IFE/IFE-coinche/src/displayRound.h File Reference	37
4.10.1 Function Documentation	38
4.10.1.1 changeCardDisplay()	38
4.10.1.2 clearCardDisplay()	38
4.10.1.3 clearContractDisplay()	39
4.10.1.4 clearDisplayedTrickPoints()	39
4.10.1.5 clearLastTrickDisplay()	39
4.10.1.6 clearTopRightBox()	39
4.10.1.7 deleteCardDisplay()	39
4.10.1.8 deleteDisplayedTrickCards()	39
4.10.1.9 deletePlayerHand()	39
4.10.1.10 displayBiddingMenuLine()	39
4.10.1.11 displayEmptyCard()	40
4.10.1.12 displayNumbersAbovePlayerHand()	40
4.10.1.13 displayPlayerHand()	40
4.10.1.14 displayPlayerName()	40
4.10.1.15 displayTrickCard()	41
4.10.1.16 prepareLastTrickDisplay()	41

4.10.1.17 preparePlayTable()	41
4.10.1.18 updateContractDisplay()	41
4.10.1.19 updateLastTrickDisplay()	42
4.10.1.20 updatePlayerTrickPoints()	42
4.10.1.21 updateRoundNbDisplay()	42
4.10.1.22 updateTeamScore()	42
4.10.1.23 updateTrickNbDisplay()	42
4.11 F:/C_IFE/IFE-coinche/src/leaderboard.c File Reference	42
4.11.1 Function Documentation	43
4.11.1.1 getTopTen()	43
4.11.1.2 increaseWins()	43
4.11.1.3 writeLine()	43
4.12 F:/C_IFE/IFE-coinche/src/leaderboard.h File Reference	44
4.12.1 Function Documentation	44
4.12.1.1 getTopTen()	44
4.12.1.2 increaseWins()	44
4.12.1.3 writeLine()	45
4.13 F:/C_IFE/IFE-coinche/src/main.c File Reference	45
4.13.1 Function Documentation	45
4.13.1.1 changePlayerSettings()	46
4.13.1.2 main()	46
4.13.1.3 mainMenu()	46
4.13.1.4 setUp()	46
4.13.1.5 tearDown()	47
4.14 F:/C_IFE/IFE-coinche/src/main.h File Reference	47
4.14.1 Function Documentation	47
4.14.1.1 changePlayerSettings()	47
4.14.1.2 mainMenu()	47
4.14.1.3 setUp()	48
4.14.1.4 tearDown()	48
4.15 F:/C_IFE/IFE-coinche/src/play.c File Reference	48
4.15.1 Function Documentation	49
4.15.1.1 awardTeamPoints()	49
4.15.1.2 bidAttempt()	49
4.15.1.3 bidUntilContract()	49
4.15.1.4 playAllGames()	50
4.15.1.5 playGame()	50
4.15.1.6 playRound()	50
4.15.1.7 playTrick()	51
4.16 F:/C_IFE/IFE-coinche/src/play.h File Reference	51
4.16.1 Function Documentation	51
4.16.1.1 awardTeamPoints()	52

4.16.1.2 bidAttempt()	52
4.16.1.3 bidUntilContract()	52
4.16.1.4 playAIGames()	53
4.16.1.5 playGame()	53
4.16.1.6 playRound()	53
4.16.1.7 playTrick()	54
4.17 F:/C_IFE/IFE-coinche/src/playerUtils.c File Reference	54
4.17.1 Function Documentation	55
4.17.1.1 cardsDistribution()	55
4.17.1.2 getPlayerCard()	55
4.17.1.3 getPlayerContract()	55
4.17.1.4 getTeamRoundPoints()	56
4.17.1.5 increaseTeamTotalScore()	56
4.18 F:/C_IFE/IFE-coinche/src/playerUtils.h File Reference	56
4.18.1 Function Documentation	57
4.18.1.1 cardsDistribution()	57
4.18.1.2 getPlayerCard()	57
4.18.1.3 getPlayerContract()	58
4.18.1.4 getTeamRoundPoints()	58
4.18.1.5 increaseTeamTotalScore()	58
4.19 F:/C_IFE/IFE-coinche/src/stringUtils.c File Reference	59
4.19.1 Macro Definition Documentation	59
4.19.1.1 UNDERLINE_SEQUENCE_LENGTH	59
4.19.2 Function Documentation	59
4.19.2.1 cropStr()	59
4.19.2.2 formatStr()	60
4.20 F:/C_IFE/IFE-coinche/src/stringUtils.h File Reference	60
4.20.1 Function Documentation	60
4.20.1.1 cropStr()	61
4.20.1.2 formatStr()	61
4.21 F:/C_IFE/IFE-coinche/src/userInput.c File Reference	61
4.21.1 Function Documentation	62
4.21.1.1 askUserCard()	62
4.21.1.2 askUserContract()	62
4.21.1.3 inputUserAcknowledgement()	62
4.21.1.4 inputUserInt()	63
4.21.1.5 inputUserName()	63
4.21.1.6 inputUserStr()	63
4.22 F:/C_IFE/IFE-coinche/src/userInput.h File Reference	64
4.22.1 Function Documentation	64
4.22.1.1 askUserCard()	64
4.22.1.2 askUserContract()	65

4.22.1.3 inputUserAcknowledgement()	65
4.22.1.4 inputUserInt()	65
4.22.1.5 inputUserName()	66
4.22.1.6 inputUserStr()	66

Index	67
--------------	-----------

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Card	5
Contract	6
Player	7

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

F:/C_IFE/IFE-coinche/src/ ai.c	9
F:/C_IFE/IFE-coinche/src/ ai.h	9
F:/C_IFE/IFE-coinche/src/ cardUtils.c	11
F:/C_IFE/IFE-coinche/src/ cardUtils.h	16
F:/C_IFE/IFE-coinche/src/ core.c	20
F:/C_IFE/IFE-coinche/src/ core.h	22
F:/C_IFE/IFE-coinche/src/ displayMain.c	29
F:/C_IFE/IFE-coinche/src/ displayMain.h	31
F:/C_IFE/IFE-coinche/src/ displayRound.c	32
F:/C_IFE/IFE-coinche/src/ displayRound.h	37
F:/C_IFE/IFE-coinche/src/ leaderboard.c	42
F:/C_IFE/IFE-coinche/src/ leaderboard.h	44
F:/C_IFE/IFE-coinche/src/ main.c	45
F:/C_IFE/IFE-coinche/src/ main.h	47
F:/C_IFE/IFE-coinche/src/ play.c	48
F:/C_IFE/IFE-coinche/src/ play.h	51
F:/C_IFE/IFE-coinche/src/ playerUtils.c	54
F:/C_IFE/IFE-coinche/src/ playerUtils.h	56
F:/C_IFE/IFE-coinche/src/ stringUtils.c	59
F:/C_IFE/IFE-coinche/src/ stringUtils.h	60
F:/C_IFE/IFE-coinche/src/ userInput.c	61
F:/C_IFE/IFE-coinche/src/ userInput.h	64

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

<i>cardsInHand[]</i>	array containing the cards in the AI's hand
<i>nbOfCardsInHand</i>	the number of cards in the array <i>cardsInHand</i>

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

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

<i>cardsInHand[]</i>	array containing the cards in the AI's hand
<i>*contract</i>	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

<i>cardsInHand[]</i>	array containing the cards in the AI's hand
<i>nbOfCardsInHand</i>	the number of cards in hand
<i>*contract</i>	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 bestTrumpStrength, Bool canPlay)
- void [findValidCardsInHand](#) (Card cardsInHand[], int nbOfCardsInHand, Card trickCards[], int nbOfTrickCards, 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

<i>cardDeck[]</i>	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

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

<i>cardArray[]</i>	array containing the cards
<i>nbOfCards</i>	how many cards are in cardArray
<i>trump</i>	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

<i>card</i>	the evaluated card
<i>trump</i>	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

<i>card</i>	the evaluated card
<i>trump</i>	the current trump
<i>roundColor</i>	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

<i>cardArray[]</i>	array containing the cards
<i>nbOfCards</i>	the number of cards in cardArray
<i>playableCards[]</i>	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

<i>cardArray[]</i>	array containing the cards to compare
<i>nbOfCards</i>	how many cards are being compared. Can be a single card
<i>trump</i>	the current trump
<i>roundColor</i>	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

<i>cardArray[]</i>	array containing the cards
<i>*nbOfCards</i>	pointer to the number of cards in cardArray
<i>cardToRemove</i>	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

<i>cardArray[]</i>	array containing the cards
<i>nbOfCards</i>	how many cards are in cardArray
<i>conditionalColor</i>	the canPlay property of each card will be set if its color is conditionalColor. Set to NULL_COLOR to bypass the condition
<i>trump</i>	the current trump. Note: this cannot be ALLTRUMP. If it is, replace ALLTRUMP with the current trick color
<i>bestTrumpStrength</i>	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
<i>canPlay</i>	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

<i>cardArray[]</i>	array containing the cards to sort
<i>nbToSort</i>	the first N cards of the array that will be sorted. Usually set to the array length to sort the whole array
<i>trump</i>	the current trump
<i>roundColor</i>	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 bestTrumpStrength, [Bool](#) canPlay)
- void [findValidCardsInHand](#) ([Card](#) cardsInHand[], int nbOfCardsInHand, [Card](#) trickCards[], int nbOfTrickCards, [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

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

<i>cardArray[]</i>	array containing the cards
<i>nbOfCards</i>	how many cards are in cardArray
<i>trump</i>	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

<i>card</i>	the evaluated card
<i>trump</i>	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

<i>card</i>	the evaluated card
<i>trump</i>	the current trump
<i>roundColor</i>	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

<i>cardArray[]</i>	array containing the cards
<i>nbOfCards</i>	the number of cards in cardArray
<i>playableCards[]</i>	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

<i>cardArray[]</i>	array containing the cards to compare
<i>nbOfCards</i>	how many cards are being compared. Can be a single card
<i>trump</i>	the current trump
<i>roundColor</i>	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

<i>cardArray[]</i>	array containing the cards
<i>*nbOfCards</i>	pointer to the number of cards in cardArray
<i>cardToRemove</i>	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

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

Parameters

<i>trump</i>	the current trump. Note: this cannot be ALLTRUMP. If it is, replace ALLTRUMP with the current trick color
<i>bestTrumpStrength</i>	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
<i>canPlay</i>	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

<i>cardArray[]</i>	array containing the cards to sort
<i>nbToSort</i>	the first N cards of the array that will be sorted. Usually set to the array length to sort the whole array
<i>trump</i>	the current trump
<i>roundColor</i>	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 * [CARD_AI_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_ALWAYSIGHTY](#) = 1, [CONTRACT_AI_STANDARD](#) = 2 }
- enum [Position](#) { [SOUTH](#) = 0, [WEST](#) = 1, [NORTH](#) = 2, [EAST](#) = 3 }
- enum [Coinche](#) { [NOT_COINCHE](#) = 0, [COINCHE](#) = 1, [OVERCOINCHE](#) = 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 * [CARD_AI_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_ALWAYSSEIGHTY	
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

<i>messageLine2</i> []	the second line of the message to display
<i>messageLine1</i> []	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

<i>nbOfLines</i>	the number of lines that should be displayed
<i>nbOfColumns</i>	the number of lines that should be displayed. Microsoft docs recommends 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

<i>messageLine2[]</i>	the second line of the message to display
<i>messageLine1[]</i>	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

<i>nbOfLines</i>	the number of lines that should be displayed
<i>nbOfColumns</i>	the number of lines that should be displayed. Microsoft docs recommends 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

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

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

Displays a line of sequential 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

<i>cardsInHand[]</i>	array containing the player's cards
<i>nbOfCardsInHand</i>	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

<i>playedCard</i>	the card to display
<i>currentPlayer</i>	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

<i>playerName</i>	the contract issuer's name
<i>contract</i>	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

<i>points</i>	the player's trick points
<i>playerPos</i>	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

<i>card</i>	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

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

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

Displays a line of sequential 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

<i>cardsInHand[]</i>	array containing the player's cards
<i>nbOfCardsInHand</i>	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

<i>playedCard</i>	the card to display
<i>currentPlayer</i>	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

<i>playerName</i>	the contract issuer's name
<i>contract</i>	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

<i>points</i>	the player's trick points
<i>playerPos</i>	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

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

<i>playerName</i> []	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

<i>*leaderboard</i>	leaderboard file pointer
<i>playerName[]</i>	name of the player
<i>wins</i>	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

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

<i>playerName[]</i>	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

<i>*leaderboard</i>	leaderboard file pointer
<i>playerName[]</i>	name of the player
<i>wins</i>	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

<i>players[]</i>	array of 4 players
<i>quitMsg[]</i>	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

<i>players[]</i>	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

<i>players[]</i>	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

<i>players[]</i>	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

<i>players[]</i>	array of 4 players
<i>quitMsg[]</i>	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

<i>players[]</i>	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

<i>players[]</i>	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

<i>players[]</i>	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 [playAllGames](#) ([Player](#) players[], int nbOfGames, int nbOfGamesWon[])

4.15.1 Function Documentation

4.15.1.1 awardTeamPoints()

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

Parameters

<i>players[]</i>	array of 4 players
<i>contract</i>	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

<i>players[]</i>	array of 4 players
<i>startingPlayer</i>	position of the starting player
<i>*contract</i>	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

<i>players[]</i>	array of 4 players
<i>startingPlayer</i>	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

<i>players[]</i>	array of 4 players
<i>nbOfGames</i>	the number of games to be played
<i>nbOfGamesWon[]</i>	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

<i>players[]</i>	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

<i>players[]</i>	array of 4 players
<i>startingPlayer</i>	position of the starting player
<i>trump</i>	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

<i>players[]</i>	array of 4 players
<i>startingPlayer</i>	position of the starting player
<i>trump</i>	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 playAllGames (Player players[], int nbOfGames, int nbOfGamesWon[])

4.16.1 Function Documentation

4.16.1.1 awardTeamPoints()

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

Parameters

<i>players[]</i>	array of 4 players
<i>contract</i>	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

<i>players[]</i>	array of 4 players
<i>startingPlayer</i>	position of the starting player
<i>*contract</i>	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

<i>players[]</i>	array of 4 players
<i>startingPlayer</i>	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

<i>players[]</i>	array of 4 players
<i>nbOfGames</i>	the number of games to be played
<i>nbOfGamesWon[]</i>	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

<i>players[]</i>	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

<i>players[]</i>	array of 4 players
<i>startingPlayer</i>	position of the starting player
<i>trump</i>	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

<i>players[]</i>	array of 4 players
<i>startingPlayer</i>	position of the starting player
<i>trump</i>	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

<i>players[]</i>	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

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

<i>player</i>	the player who has to make a decision
<i>*contract</i>	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

<i>players[]</i>	array of 4 players
<i>player</i>	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

<i>players[]</i>	array of 4 players
<i>player</i>	position of a player in the team of interest
<i>roundScore</i>	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

<i>players[]</i>	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

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

<i>player</i>	the player who has to make a decision
<i>*contract</i>	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

<i>players[]</i>	array of 4 players
<i>player</i>	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

<i>players[]</i>	array of 4 players
<i>player</i>	position of a player in the team of interest
<i>roundScore</i>	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

<i>string[]</i>	the string to crop
<i>maxLength</i>	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

<i>string[]</i>	the string to center
<i>maxLength</i>	the maximum length of the space the centered string will be displayed in
<i>textPosition</i>	text alignment within the string (left, center, right)
<i>underline</i>	TRUE if the text could 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

<i>string[]</i>	the string to crop
<i>maxLength</i>	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

<i>string[]</i>	the string to center
<i>maxLength</i>	the maximum length of the space the centered string will be displayed in
<i>textPosition</i>	text alignment within the string (left, center, right)
<i>underline</i>	TRUE if the text could 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

<i>cardArray[]</i>	array containing the player's cards
<i>nbOfCards</i>	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

<i>*contract</i>	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

<i>displayMsg[]</i>	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

<i>minBound</i>	the minimum valid value the user can enter
<i>maxBound</i>	the maximum valid value the user can enter
<i>displayStr[]</i>	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

<i>displayMsg[]</i>	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

<i>maxStrLength</i>	the maximum length of the input string, including the terminating \0. Anything bigger than that will be cropped
<i>displayStrline1[]</i>	first line of the prompt given to the user asking for an input
<i>displayStrline2[]</i>	second line of the prompt given to the user. If useSecondLineAsInput is set to true, this is ignored
<i>useSecondLineAsInput</i>	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

<i>cardArray[]</i>	array containing the player's cards
<i>nbOfCards</i>	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

<i>*contract</i>	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

<i>displayMsg[]</i>	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

<i>minBound</i>	the minimum valid value the user can enter
<i>maxBound</i>	the maximum valid value the user can enter
<i>displayStr[]</i>	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

<i>displayMsg[]</i>	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

<i>maxStrLength</i>	the maximum length of the input string, including the terminating \0. Anything bigger than that will be cropped
<i>displayStrline1[]</i>	first line of the prompt given to the user asking for an input
<i>displayStrline2[]</i>	second line of the prompt given to the user. If useSecondLineAsInput is set to true, this is ignored
<i>useSecondLineAsInput</i>	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

ACE

core.h, [28](#)

ai.h

getAICardFirstAvailable, [9](#)
getAICardStandard, [10](#)
getAIContractAlwaysEighty, [10](#)
getAIContractStandard, [11](#)

ALLTRUMP

core.h, [26](#)

askUserCard

userInput.c, [62](#)
userInput.h, [64](#)

askUserContract

userInput.c, [62](#)
userInput.h, [65](#)

awardTeamPoints

play.c, [49](#)
play.h, [51](#)

bidAttempt

play.c, [49](#)
play.h, [52](#)

bidUntilContract

play.c, [49](#)
play.h, [52](#)

Bool

core.h, [24](#), [25](#)

canPlay

Card, [5](#)

CAPOT

core.h, [27](#)

Card, [5](#)

canPlay, [5](#)
color, [5](#)
core.h, [24](#)
value, [5](#)

CARD_AI_FIRSTAVAILABLE

core.h, [26](#)

CARD_AI_STANDARD

core.h, [26](#)

CARD_POINTS_TABLE

core.c, [21](#)
core.h, [28](#)

CARD_USER

core.h, [26](#)

CardAI

core.h, [24](#), [26](#)

cardAI

Player, [7](#)

CARDAI_STR_TABLE

core.c, [21](#)
core.h, [28](#)

cards

Player, [7](#)

cardsDistribution

playerUtils.c, [55](#)
playerUtils.h, [57](#)

cardUtils.c

createDeck, [12](#)
findValidCardsInHand, [12](#)
getCardArrayPoints, [12](#)
getCardPoints, [13](#)
getCardStrength, [13](#)
getPlayableCards, [13](#)
getStrongestCard, [14](#)
removeCard, [14](#)
setCanPlay, [15](#)
sortCards, [15](#)

cardUtils.h

createDeck, [16](#)
findValidCardsInHand, [16](#)
getCardArrayPoints, [17](#)
getCardPoints, [17](#)
getCardStrength, [17](#)
getPlayableCards, [18](#)
getStrongestCard, [18](#)
removeCard, [19](#)
setCanPlay, [19](#)
sortCards, [20](#)

changeCardDisplay

displayRound.c, [33](#)
displayRound.h, [38](#)

changePlayerSettings

main.c, [45](#)
main.h, [47](#)

clearCardDisplay

displayRound.c, [33](#)
displayRound.h, [38](#)

clearContractDisplay

displayRound.c, [33](#)
displayRound.h, [38](#)

clearDisplayedTrickPoints

displayRound.c, [34](#)
displayRound.h, [39](#)

clearInfoMsg

displayMain.c, [29](#)
displayMain.h, [31](#)

clearLastTrickDisplay

- displayRound.c, [34](#)
- displayRound.h, [39](#)
- clearTopRightBox
 - displayRound.c, [34](#)
 - displayRound.h, [39](#)
- CLUB
 - core.h, [26](#)
- Coinche
 - core.h, [24](#), [26](#)
- coinche
 - Contract, [6](#)
- COINCHE_STR_TABLE
 - core.c, [21](#)
 - core.h, [28](#)
- COINCHED
 - core.h, [26](#)
- Color
 - core.h, [24](#), [26](#)
- color
 - Card, [5](#)
- COLOR_STR_TABLE
 - core.c, [21](#)
 - core.h, [28](#)
- Contract, [6](#)
 - coinche, [6](#)
 - core.h, [24](#)
 - issuer, [6](#)
 - points, [6](#)
 - trump, [6](#)
 - type, [6](#)
- CONTRACT_AI_ALWAYSSEIGHTY
 - core.h, [27](#)
- CONTRACT_AI_STANDARD
 - core.h, [27](#)
- CONTRACT_USER
 - core.h, [27](#)
- ContractAI
 - core.h, [25](#), [26](#)
- contractAI
 - Player, [7](#)
- CONTRACTAI_STR_TABLE
 - core.c, [21](#)
 - core.h, [28](#)
- ContractType
 - core.h, [25](#), [27](#)
- CONTRACTTYPE_STR_TABLE
 - core.c, [22](#)
 - core.h, [29](#)
- core.c
 - CARD_POINTS_TABLE, [21](#)
 - CARDAI_STR_TABLE, [21](#)
 - COINCHE_STR_TABLE, [21](#)
 - COLOR_STR_TABLE, [21](#)
 - CONTRACTAI_STR_TABLE, [21](#)
 - CONTRACTTYPE_STR_TABLE, [22](#)
 - VALUE_STR_TABLE, [22](#)
- core.h
 - ACE, [28](#)
 - ALLTRUMP, [26](#)
 - Bool, [24](#), [25](#)
 - CAPOT, [27](#)
 - Card, [24](#)
 - CARD_AI_FIRSTAVAILABLE, [26](#)
 - CARD_AI_STANDARD, [26](#)
 - CARD_POINTS_TABLE, [28](#)
 - CARD_USER, [26](#)
 - CardAI, [24](#), [26](#)
 - CARDAI_STR_TABLE, [28](#)
 - CLUB, [26](#)
 - Coinche, [24](#), [26](#)
 - COINCHE_STR_TABLE, [28](#)
 - COINCHED, [26](#)
 - Color, [24](#), [26](#)
 - COLOR_STR_TABLE, [28](#)
 - Contract, [24](#)
 - CONTRACT_AI_ALWAYSSEIGHTY, [27](#)
 - CONTRACT_AI_STANDARD, [27](#)
 - CONTRACT_USER, [27](#)
 - ContractAI, [25](#), [26](#)
 - CONTRACTAI_STR_TABLE, [28](#)
 - ContractType, [25](#), [27](#)
 - CONTRACTTYPE_STR_TABLE, [29](#)
 - DIAMOND, [26](#)
 - EAST, [27](#)
 - EIGHT, [28](#)
 - FALSE, [26](#)
 - GENERAL, [27](#)
 - HEART, [26](#)
 - JACK, [28](#)
 - KING, [28](#)
 - MAX_PLAYER_NAME_LENGTH, [23](#)
 - NB_CARD_AI, [23](#)
 - NB_CONTRACT_AI, [24](#)
 - NINE, [28](#)
 - NORTH, [27](#)
 - NOT_COINCHED, [26](#)
 - NOTRUMP, [26](#)
 - NULL_COLOR, [26](#)
 - NULL_VALUE, [28](#)
 - OVERCOINCHED, [26](#)
 - Player, [25](#)
 - POINTS, [27](#)
 - Position, [25](#), [27](#)
 - QUEEN, [28](#)
 - SEVEN, [28](#)
 - SOUTH, [27](#)
 - SPADE, [26](#)
 - TEN, [28](#)
 - TEXT_CENTER, [27](#)
 - TEXT_LEFT, [27](#)
 - TEXT_RIGHT, [27](#)
 - TextPosition, [25](#), [27](#)
 - TRUE, [26](#)
 - Value, [25](#), [28](#)
 - VALUE_STR_TABLE, [29](#)
 - WEST, [27](#)

- createDeck
 - cardUtils.c, 12
 - cardUtils.h, 16
- croppedName
 - Player, 7
- cropStr
 - stringUtils.c, 59
 - stringUtils.h, 60
- deleteCardDisplay
 - displayRound.c, 34
 - displayRound.h, 39
- deleteDisplayedTrickCards
 - displayRound.c, 34
 - displayRound.h, 39
- deletePlayerHand
 - displayRound.c, 34
 - displayRound.h, 39
- DIAMOND
 - core.h, 26
- displayBiddingMenuLine
 - displayRound.c, 34
 - displayRound.h, 39
- displayCredits
 - displayMain.c, 29
 - displayMain.h, 31
- displayEmptyCard
 - displayRound.c, 35
 - displayRound.h, 40
- displayFrame
 - displayMain.c, 30
 - displayMain.h, 31
- displayInfoMsg
 - displayMain.c, 30
 - displayMain.h, 31
- displayLeaderboard
 - displayMain.c, 30
 - displayMain.h, 32
- displayMain.c
 - clearInfoMsg, 29
 - displayCredits, 29
 - displayFrame, 30
 - displayInfoMsg, 30
 - displayLeaderboard, 30
 - displayMenu, 30
 - resizeCmdWindow, 30
- displayMain.h
 - clearInfoMsg, 31
 - displayCredits, 31
 - displayFrame, 31
 - displayInfoMsg, 31
 - displayLeaderboard, 32
 - displayMenu, 32
 - resizeCmdWindow, 32
- displayMenu
 - displayMain.c, 30
 - displayMain.h, 32
- displayNumbersAbovePlayerHand
 - displayRound.c, 35
- displayRound.h, 40
- displayPlayerHand
 - displayRound.c, 35
 - displayRound.h, 40
- displayPlayerName
 - displayRound.c, 35
 - displayRound.h, 40
- displayRound.c
 - changeCardDisplay, 33
 - clearCardDisplay, 33
 - clearContractDisplay, 33
 - clearDisplayedTrickPoints, 34
 - clearLastTrickDisplay, 34
 - clearTopRightBox, 34
 - deleteCardDisplay, 34
 - deleteDisplayedTrickCards, 34
 - deletePlayerHand, 34
 - displayBiddingMenuLine, 34
 - displayEmptyCard, 35
 - displayNumbersAbovePlayerHand, 35
 - displayPlayerHand, 35
 - displayPlayerName, 35
 - displayTrickCard, 36
 - prepareLastTrickDisplay, 36
 - preparePlayTable, 36
 - updateContractDisplay, 36
 - updateLastTrickDisplay, 36
 - updatePlayerTrickPoints, 37
 - updateRoundNbDisplay, 37
 - updateTeamScore, 37
 - updateTrickNbDisplay, 37
- displayRound.h
 - changeCardDisplay, 38
 - clearCardDisplay, 38
 - clearContractDisplay, 38
 - clearDisplayedTrickPoints, 39
 - clearLastTrickDisplay, 39
 - clearTopRightBox, 39
 - deleteCardDisplay, 39
 - deleteDisplayedTrickCards, 39
 - deletePlayerHand, 39
 - displayBiddingMenuLine, 39
 - displayEmptyCard, 40
 - displayNumbersAbovePlayerHand, 40
 - displayPlayerHand, 40
 - displayPlayerName, 40
 - displayTrickCard, 41
 - prepareLastTrickDisplay, 41
 - preparePlayTable, 41
 - updateContractDisplay, 41
 - updateLastTrickDisplay, 41
 - updatePlayerTrickPoints, 42
 - updateRoundNbDisplay, 42
 - updateTeamScore, 42
 - updateTrickNbDisplay, 42
- displayTrickCard
 - displayRound.c, 36
 - displayRound.h, 41

- EAST
 - core.h, [27](#)
- EIGHT
 - core.h, [28](#)
- F:/C_IFE/IFE-coinche/src/ai.c, [9](#)
- F:/C_IFE/IFE-coinche/src/ai.h, [9](#)
- F:/C_IFE/IFE-coinche/src/cardUtils.c, [11](#)
- F:/C_IFE/IFE-coinche/src/cardUtils.h, [16](#)
- F:/C_IFE/IFE-coinche/src/core.c, [20](#)
- F:/C_IFE/IFE-coinche/src/core.h, [22](#)
- F:/C_IFE/IFE-coinche/src/displayMain.c, [29](#)
- F:/C_IFE/IFE-coinche/src/displayMain.h, [31](#)
- F:/C_IFE/IFE-coinche/src/displayRound.c, [32](#)
- F:/C_IFE/IFE-coinche/src/displayRound.h, [37](#)
- F:/C_IFE/IFE-coinche/src/leaderboard.c, [42](#)
- F:/C_IFE/IFE-coinche/src/leaderboard.h, [44](#)
- F:/C_IFE/IFE-coinche/src/main.c, [45](#)
- F:/C_IFE/IFE-coinche/src/main.h, [47](#)
- F:/C_IFE/IFE-coinche/src/play.c, [48](#)
- F:/C_IFE/IFE-coinche/src/play.h, [51](#)
- F:/C_IFE/IFE-coinche/src/playerUtils.c, [54](#)
- F:/C_IFE/IFE-coinche/src/playerUtils.h, [56](#)
- F:/C_IFE/IFE-coinche/src/stringUtils.c, [59](#)
- F:/C_IFE/IFE-coinche/src/stringUtils.h, [60](#)
- F:/C_IFE/IFE-coinche/src/userInput.c, [61](#)
- F:/C_IFE/IFE-coinche/src/userInput.h, [64](#)
- FALSE
 - core.h, [26](#)
- findValidCardsInHand
 - cardUtils.c, [12](#)
 - cardUtils.h, [16](#)
- formatStr
 - stringUtils.c, [60](#)
 - stringUtils.h, [61](#)
- GENERAL
 - core.h, [27](#)
- getAICardFirstAvailable
 - ai.h, [9](#)
- getAICardStandard
 - ai.h, [10](#)
- getAIContractAlwaysEighty
 - ai.h, [10](#)
- getAIContractStandard
 - ai.h, [11](#)
- getCardArrayPoints
 - cardUtils.c, [12](#)
 - cardUtils.h, [17](#)
- getCardPoints
 - cardUtils.c, [13](#)
 - cardUtils.h, [17](#)
- getCardStrength
 - cardUtils.c, [13](#)
 - cardUtils.h, [17](#)
- getPlayableCards
 - cardUtils.c, [13](#)
 - cardUtils.h, [18](#)
- getPlayerCard
 - playerUtils.c, [55](#)
 - playerUtils.h, [57](#)
- getPlayerContract
 - playerUtils.c, [55](#)
 - playerUtils.h, [58](#)
- getStrongestCard
 - cardUtils.c, [14](#)
 - cardUtils.h, [18](#)
- getTeamRoundPoints
 - playerUtils.c, [56](#)
 - playerUtils.h, [58](#)
- getTopTen
 - leaderboard.c, [43](#)
 - leaderboard.h, [44](#)
- HEART
 - core.h, [26](#)
- increaseTeamTotalScore
 - playerUtils.c, [56](#)
 - playerUtils.h, [58](#)
- increaseWins
 - leaderboard.c, [43](#)
 - leaderboard.h, [44](#)
- inputUserAcknowledgement
 - userInput.c, [62](#)
 - userInput.h, [65](#)
- inputUserInt
 - userInput.c, [63](#)
 - userInput.h, [65](#)
- inputUserName
 - userInput.c, [63](#)
 - userInput.h, [65](#)
- inputUserStr
 - userInput.c, [63](#)
 - userInput.h, [66](#)
- issuer
 - Contract, [6](#)
- JACK
 - core.h, [28](#)
- KING
 - core.h, [28](#)
- leaderboard.c
 - getTopTen, [43](#)
 - increaseWins, [43](#)
 - writeLine, [43](#)
- leaderboard.h
 - getTopTen, [44](#)
 - increaseWins, [44](#)
 - writeLine, [45](#)
- main
 - main.c, [46](#)
- main.c
 - changePlayerSettings, [45](#)
 - main, [46](#)
 - mainMenu, [46](#)

- setUp, 46
 - tearDown, 46
- main.h
 - changePlayerSettings, 47
 - mainMenu, 47
 - setUp, 48
 - tearDown, 48
- mainMenu
 - main.c, 46
 - main.h, 47
- MAX_PLAYER_NAME_LENGTH
 - core.h, 23
- name
 - Player, 7
- NB_CARD_AI
 - core.h, 23
- NB_CONTRACT_AI
 - core.h, 24
- nbOfCards
 - Player, 8
- NINE
 - core.h, 28
- NORTH
 - core.h, 27
- NOT_COINCHED
 - core.h, 26
- NOTRUMP
 - core.h, 26
- NULL_COLOR
 - core.h, 26
- NULL_VALUE
 - core.h, 28
- OVERCOINCHED
 - core.h, 26
- play.c
 - awardTeamPoints, 49
 - bidAttempt, 49
 - bidUntilContract, 49
 - playAIGames, 50
 - playGame, 50
 - playRound, 50
 - playTrick, 51
- play.h
 - awardTeamPoints, 51
 - bidAttempt, 52
 - bidUntilContract, 52
 - playAIGames, 53
 - playGame, 53
 - playRound, 53
 - playTrick, 54
- playAIGames
 - play.c, 50
 - play.h, 53
- Player, 7
 - cardAI, 7
 - cards, 7
 - contractAI, 7
 - core.h, 25
 - croppedName, 7
 - name, 7
 - nbOfCards, 8
 - pos, 8
 - score, 8
 - teamScore, 8
- playerUtils.c
 - cardsDistribution, 55
 - getPlayerCard, 55
 - getPlayerContract, 55
 - getTeamRoundPoints, 56
 - increaseTeamTotalScore, 56
- playerUtils.h
 - cardsDistribution, 57
 - getPlayerCard, 57
 - getPlayerContract, 58
 - getTeamRoundPoints, 58
 - increaseTeamTotalScore, 58
- playGame
 - play.c, 50
 - play.h, 53
- playRound
 - play.c, 50
 - play.h, 53
- playTrick
 - play.c, 51
 - play.h, 54
- POINTS
 - core.h, 27
- points
 - Contract, 6
- pos
 - Player, 8
- Position
 - core.h, 25, 27
- prepareLastTrickDisplay
 - displayRound.c, 36
 - displayRound.h, 41
- preparePlayTable
 - displayRound.c, 36
 - displayRound.h, 41
- QUEEN
 - core.h, 28
- removeCard
 - cardUtils.c, 14
 - cardUtils.h, 19
- resizeCmdWindow
 - displayMain.c, 30
 - displayMain.h, 32
- score
 - Player, 8
- setCanPlay
 - cardUtils.c, 15
 - cardUtils.h, 19

- setUp
 - main.c, [46](#)
 - main.h, [48](#)
- SEVEN
 - core.h, [28](#)
- sortCards
 - cardUtils.c, [15](#)
 - cardUtils.h, [20](#)
- SOUTH
 - core.h, [27](#)
- SPADE
 - core.h, [26](#)
- stringUtils.c
 - cropStr, [59](#)
 - formatStr, [60](#)
 - UNDERLINE_SEQUENCE_LENGTH, [59](#)
- stringUtils.h
 - cropStr, [60](#)
 - formatStr, [61](#)
- teamScore
 - Player, [8](#)
- tearDown
 - main.c, [46](#)
 - main.h, [48](#)
- TEN
 - core.h, [28](#)
- TEXT_CENTER
 - core.h, [27](#)
- TEXT_LEFT
 - core.h, [27](#)
- TEXT_RIGHT
 - core.h, [27](#)
- TextPosition
 - core.h, [25](#), [27](#)
- TRUE
 - core.h, [26](#)
- trump
 - Contract, [6](#)
- type
 - Contract, [6](#)
- UNDERLINE_SEQUENCE_LENGTH
 - stringUtils.c, [59](#)
- updateContractDisplay
 - displayRound.c, [36](#)
 - displayRound.h, [41](#)
- updateLastTrickDisplay
 - displayRound.c, [36](#)
 - displayRound.h, [41](#)
- updatePlayerTrickPoints
 - displayRound.c, [37](#)
 - displayRound.h, [42](#)
- updateRoundNbDisplay
 - displayRound.c, [37](#)
 - displayRound.h, [42](#)
- updateTeamScore
 - displayRound.c, [37](#)
 - displayRound.h, [42](#)
- updateTrickNbDisplay
 - displayRound.c, [37](#)
 - displayRound.h, [42](#)
- userInput.c
 - askUserCard, [62](#)
 - askUserContract, [62](#)
 - inputUserAcknowledgement, [62](#)
 - inputUserInt, [63](#)
 - inputUserName, [63](#)
 - inputUserStr, [63](#)
- userInput.h
 - askUserCard, [64](#)
 - askUserContract, [65](#)
 - inputUserAcknowledgement, [65](#)
 - inputUserInt, [65](#)
 - inputUserName, [65](#)
 - inputUserStr, [66](#)
- Value
 - core.h, [25](#), [28](#)
- value
 - Card, [5](#)
- VALUE_STR_TABLE
 - core.c, [22](#)
 - core.h, [29](#)
- WEST
 - core.h, [27](#)
- writeLine
 - leaderboard.c, [43](#)
 - leaderboard.h, [45](#)