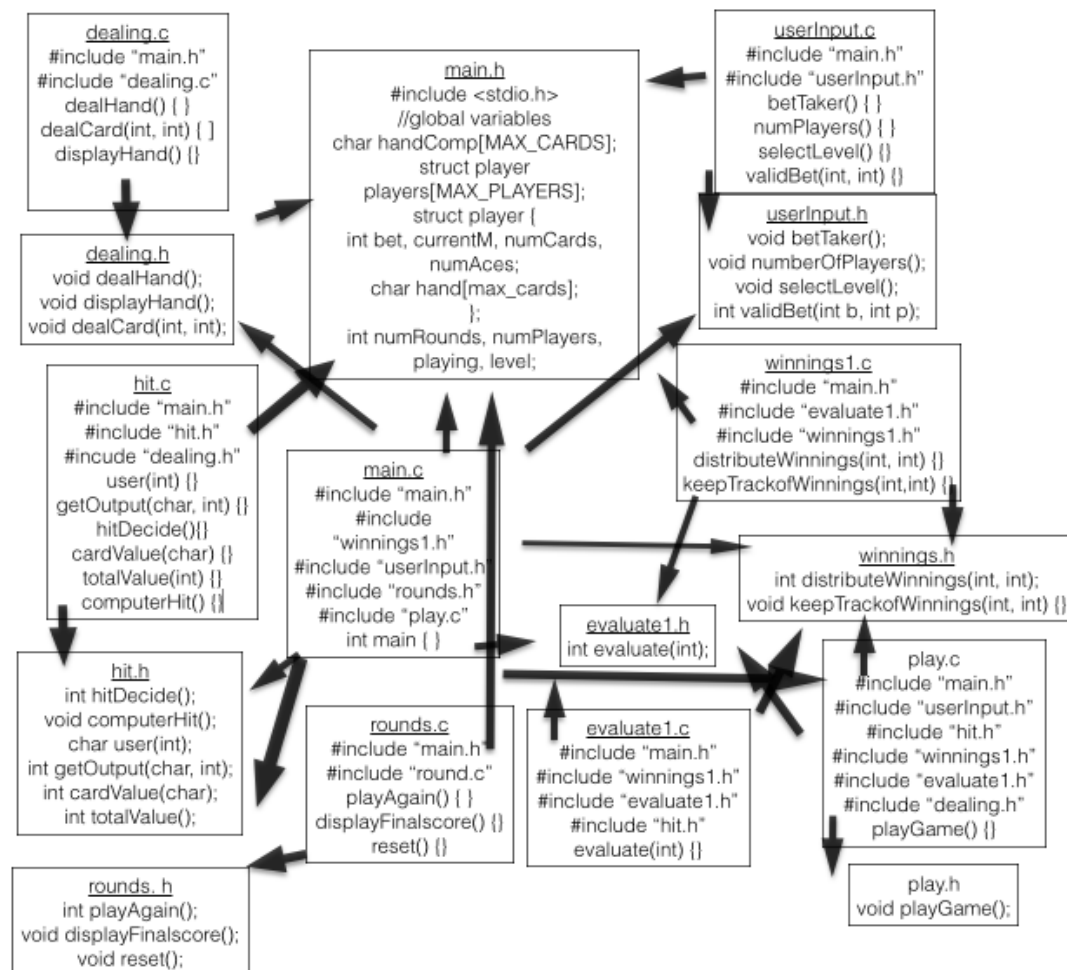


Programming Paradigms Final Project- Blackjack Design

Grace Baelen-King and Angela Mastrianni

April 28th, 2014



Rules and Scoring

1. The users input the number of players. Each player and the dealer are dealt two cards. The players also make their bets at this time.
2. After the cards are dealt, the dealer goes clockwise around to the players and asks them if they would like to stay or hit.
3. If they player chooses to hit, they recieve anther card face up. After recieving that card, they can choose if they would like to hit again or stay. This continues until the player finally chooses to stay or goes over twenty-one (which is called getting a bust).
4. After the dealer handles all of the players, the dealer must choose to give a card(s) to themselves or stay. In tradition blackjack, if the dealer has under 17 points, they are required to hit. If they are at 17 points or above, they must stay. If the player wishes to play an easier game of blackjack, they computer must stay whenever they are at 15 points or above.
5. Once the players and dealers have their cards, they must compare to determine who wins or loses money. The players are only competing against the dealer and not each other.
6. If a player has more points than the dealer without busting, the player recieves three times the money that they bet. If a player has less points then the dealer, they lose twice the amount of the money that they bet. If the player and dealer have the same amount of points, the player gets to keep their bet. If both the player and the dealer bust, the player gets to keep their bet.
7. The players then get to choose if they wish to play another round. If they do not choose to play another round, the game statistics will be printed out.

Files and their Functions

1. UserInput.c
 - void numPlayers() : This function first asks the user how many players they would like to have (up to five). Then it creates the specified number of players and stores the players in a global array. The function error checks to ensure that the number of players is correct.
 - void selectLevel(): this function allows the users to choose which level they would to play at (0 being easy and 1 being hard). It error checks, then updates the global level variable.

- void betTaker(): The the function takes in and stores the player's bet.
- int validBet(int, int): this functions checks to make sure the bet does not exceed the player's maximum amount of money. It takes in the inputted bet and the specific player. It returns an integer which specifies if the bet was correct or not.

2. Dealing.c

- void dealCard(int, int): The dealHand function takes in the amount of cards to be dealt and the player/computer to whom they should be dealt. The cards will be stored in the global player structure or in the global computer hand.
- void dealHand(): gives the computer and players their initial cards.
- void displayHand(): this function will print out both cards that each player has been dealt. It will also print out one of the computer's cards.

3. Hit.c

- char user(int): this function takes in the user id and handles the interactions with the player. It returns the player's answer to wheter or not they would like to hit (either a y, n or c).
- int getOutput(char, int): this function takes in the result from the user function and the player id. It tells the user the computer's second card if they input the cheat code and then calls the user and getOutput functions again. Once the player answers y or n, it returns a 1 for a y and a 0 for a n.
- int hitDecide() This function calls the user function and handles the output from the getOutput function. It returns the result to main so that main can give the player another card if necessary.
- void computerHit() : The dealer has certain rules for when it must hit. This function will decide if the computer should hit and then give it another card if it does need to hit. If the global variable level is 1, a different set of rules will be used to change the level of the game. If the computer gets another hit, the function will alter the global variable char handComp.
- int cardValue(char c): this function takes in the character representing the value of the card. It calculates this value and returns its integer value.
- int totalValue(): this function calculates the total value of each hand. It also accounts for aces and adjusts their values, if necessary. It returns the total.

4. Evaluate1.c

- `int evaluate(int j)`: this function takes in the player id and compares the player's total to the computer's total. It returns a 0 if the player loses, a 1 if the player wins and a 2 if it is a draw.

5. Winnings1.c

- `int distributeWinnings(int, int)`: this function takes in the integer from the `compareFinalhands` function which indicates if the dealer or the player won and the player id. It calculates the appropriate winnings based on the initial bets which can be found in each player's global structure. It then prints out the amount that the player won or lost and returns that value.
- `void keepTrackofWinnings(int, int)`: this function takes in the amount of winnings from the previous function and the player id. It then modifies these player's current money (the `currentM` part of the structure) using the amount that they player won or lost.

6. Rounds.c

- `int playAgain()` At the end of a round this function would ask the human players if they would like to play another round of Blackjack. Returns a 0 if the players say no, returns 1 if they say yes.
- `displayFinalScore()`: If the human players chose not to play another hand in `playAgain`, this function would display the final amount of money the players had each accumulated.
- `void reset()` : if another round is to be played, this function resets all the appropriate values in order to restart the game.