**Project 2**

Tittle

**Three Card Game**

Course

**CSC-17A**

Section

**48096**

Due Date

**December 9, 2016**

Author

**Sili Guo**

**Table of Contents**

1. **Introduction………………………………………………3**
2. **Game rules ………………………………………………4**
   1. **Decide Players Number……………………………….4**
   2. **Discard…………………………………………………4**
   3. **Player’s Action…………………………………………5**
   4. **Too See………………………………………………….5**
   5. **Raise Bet………………………………………………..6**
   6. **Challenge Previous Player…………………………….6**
   7. **Compare Rules…………………………………………7**
   8. **Result……………………………………………………8**
3. **Summary .…………………………………………………9**
4. **Challenge During Program………………………………10**
   1. **Logic Errors……………………………………………10**
   2. **How to Control AI Decision …………………………..10**
   3. **Conversion Between integer and character…………..11**
5. **Description ………………………………………………..12**
   1. **Sample Input / Output…………………………………12**
   2. **Flowcharts………………………………………………14**
   3. **Pseudocode ……………………………………………..16**
   4. **Variables ………………………………………………..19**
   5. **Concepts ………………………………………………..20**
6. **UML ………………………………………………………..22**
7. **Introduction**

Three Card Game is a traditional poker gabbling game that is popular in China. Its rules is similar to many of western card games, but it is more flexible about the number of people playing this game. And the game between two people and several people can experience different feeling, and use different strategies during the game.

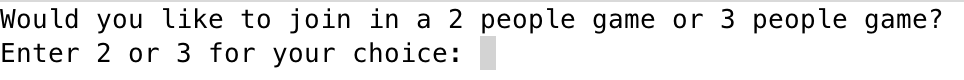
I like poker games because I enjoy the feeling of guessing others’ cards by the action they do during the game; the process of holding small cards to try to bluff makes me feel excited. However, after playing many western poker games, such as Texas hold’em, and Black jack, I miss the day I played Three Card Game with my Chinese friends, for it makes me feel more about relax, rather than nervous.

The most interesting part for this game is there’s always a way to win. The rules of the smallest cards set of 2, 3, and 5 can beat the largest cards set of full house! You will never know if you can win others or not before open both cards sets.

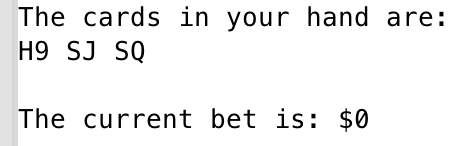
Note: In this game, 0 represent to 10.

**II. Game Rules**

**A. Decide players number**

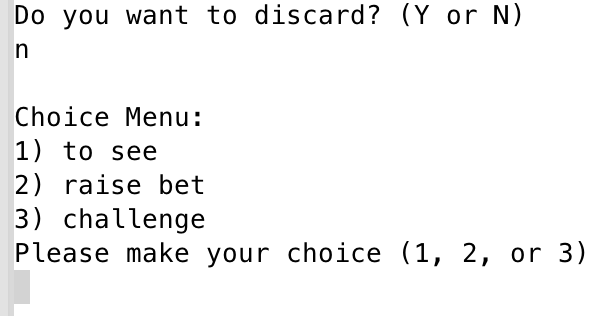
The Three Card Game Program only limit 2~3 players to play this game. At the beginning, the program will ask player if he wants to join a 2 people game, or 3 people game

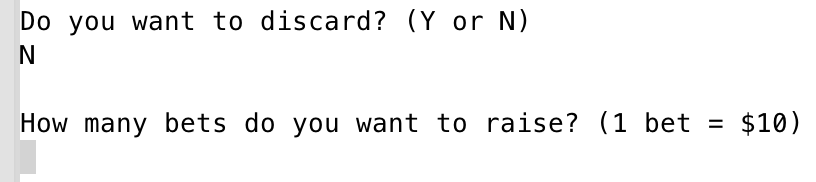
**B. Discard**

After choose the number of players, the game started. Player can see the cards in his hand, and the bet of the table currently.

Game starts by randomly pick the dealer. At the beginning of each round, the system will ask player if he would like to discard his cards. If the player was luckily picked as the dealer, then he will has one chance to leave the table without paying anything; otherwise, player can not take the bet on the table back.

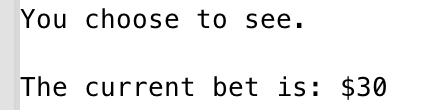
**C. Player’s Action**

If the player decide to play the game with his cards, the system will automatically display the choice menu of player’s action: to see, raise bet, or challenge previous player.

Note: During the first round of the game, if the dealer did not discard, he can only choose to raise bet.

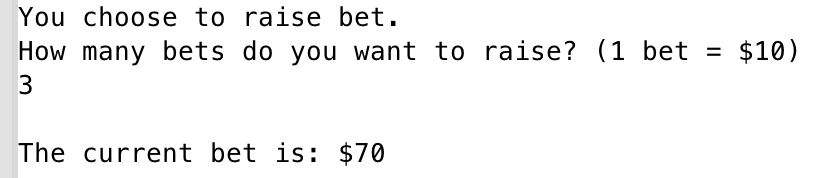
**D. To See**

To see is a simply strategy in card games to see how other people react, which may implicit if he has a good cards set or not. By choosing to see, the player’s bet will be equal to the current bet on the table, and the current bet will not change.



**E. Raise Bet**

If a player choose to raise bet, that means he wants to increase the amount of the bet on the table. This can be regarding as a signal of good cards set; but also can be a good strategy of bluff during the game.

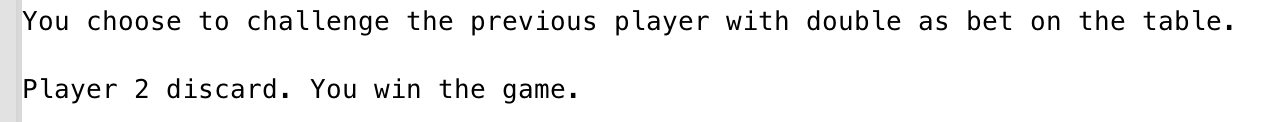
To raise bet, there will always be a minimum and maximum amount for the bet to raise. For this program, player can raise at least 1 bet($10), but at most 10 bets($100) for each round; and the table can hold totally 120 bets ($1200);

**F. Challenge Previous Player**

The most interesting part of this game is to enjoy the process of challenging others and waiting to be challenged.

Challenge in Three Card Game only allow two players to compare(the compare rules will be discussed under this section); and there will be a judge to look at both players’ card and report the winner, which means a player will not know others card even his challenge success.

When a player decide to challenge the previous player, he must has at least twice as much money as the bet of the table currently, because challenging others will cost twice as much as bet for their private bet. The winner takes all money, and loser exit game.

When a player is challenged, he can choose to discard and give the bet on the table to the challenger, or he can choose to bet the same amount of challenger to compare their cards.

**G. Compare Rules**

There are seven ranks of cards set in Three Card Game:

* + Full House (three same cards)
  + Straight Flush
  + Flush
  + Straight
  + One pair
  + Single
  + Special (2, 3, 5 in dif suit)

The rules are simple: full house > straight flush > flush > straight > one pair > single > special > full house.

Remind that special is the smallest card set in the game, but it can beat full house.

If two players have cards in same rank, then compare from the largest card to smallest one.

Since Three Card Game do not count suit, if two players have exactly the same cards, the challenger is the loser.

**H. Result**

If the player lose during the challenge, the game is over; otherwise, output how much the player won or lost.

**III. Summary**

|  |  |
| --- | --- |
| **Total Line of Code** | **1000+** |
| **Comment Line** |  |
| **Variable** |  |
| **Function** |  |

Note: Some of the comment line are used for testing.

The total function based on the class Player. I stored a single card’s information in Card class, and derive another class called Player, which stores a Card pointer, the rank of card set, the money and bet of player. To make sure the data is correct, I write it into a binary file, and read it when necessary. The program is constructed with decision makings. I used lots of loop and switch case to help program decide the branches that game will go. The AI makes decisions based on the random number creation. To pass variables between main and function, I use pointer point to function, pass by address, and pass pointer as variables. The variables are called by class in function that make sure the number wont’t be mistakenly changed easily.

**VI. Challenge During Program**

**A. Logic Errors**

The two classes and pointers make me a little confused about the logic, and I made some mistakes about my pointer in class, and about the way I called get function, which cost me really long time to work on.

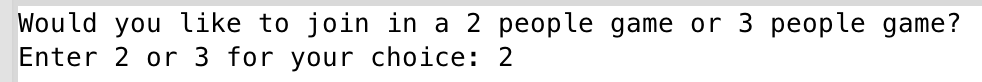
**B. Transfer Program From Structure**

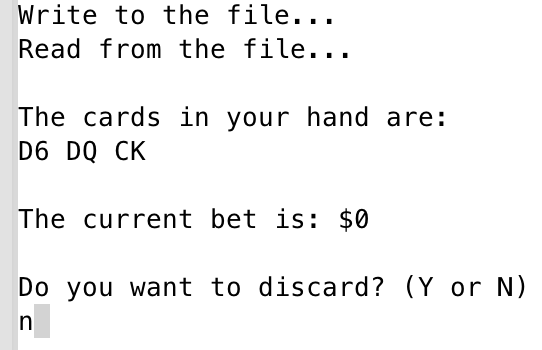
Although structure is so similar as class, the way we use the variables declared in structure and class are so different, that one we can called just by pointer, one ask to call function to get the value.

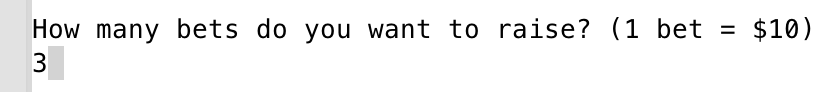
I declared a pointer inside Player class, and return the pointer to my main, however, it is hard to get the value inside the pointer, I finally has to create 3 separate pointers to contain the pointer which makes me change a lot in my program.

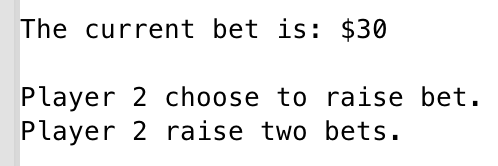
**V. Description**

**A. Sample Input/Output**

First decide the players number

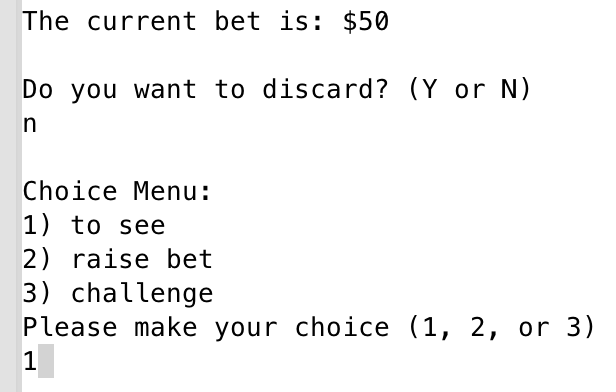
The player is luckily chose as dealer. Ask if player if he would like to discard

For first round, only ask to raise bet

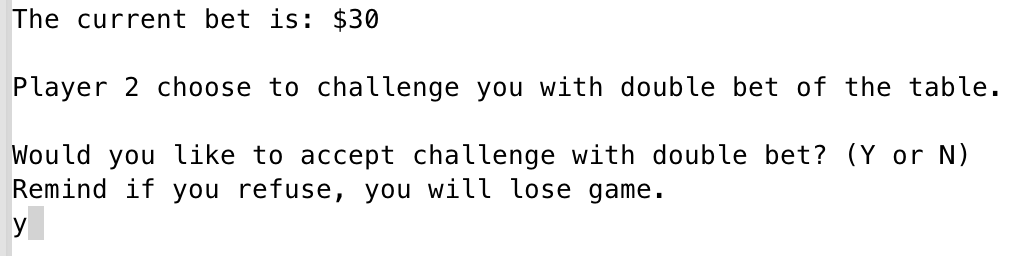
Then round for AI

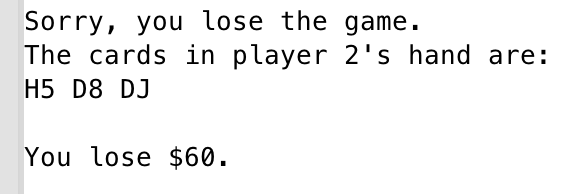
Player’s 2nd round, ask if discard

If no, output choice menu

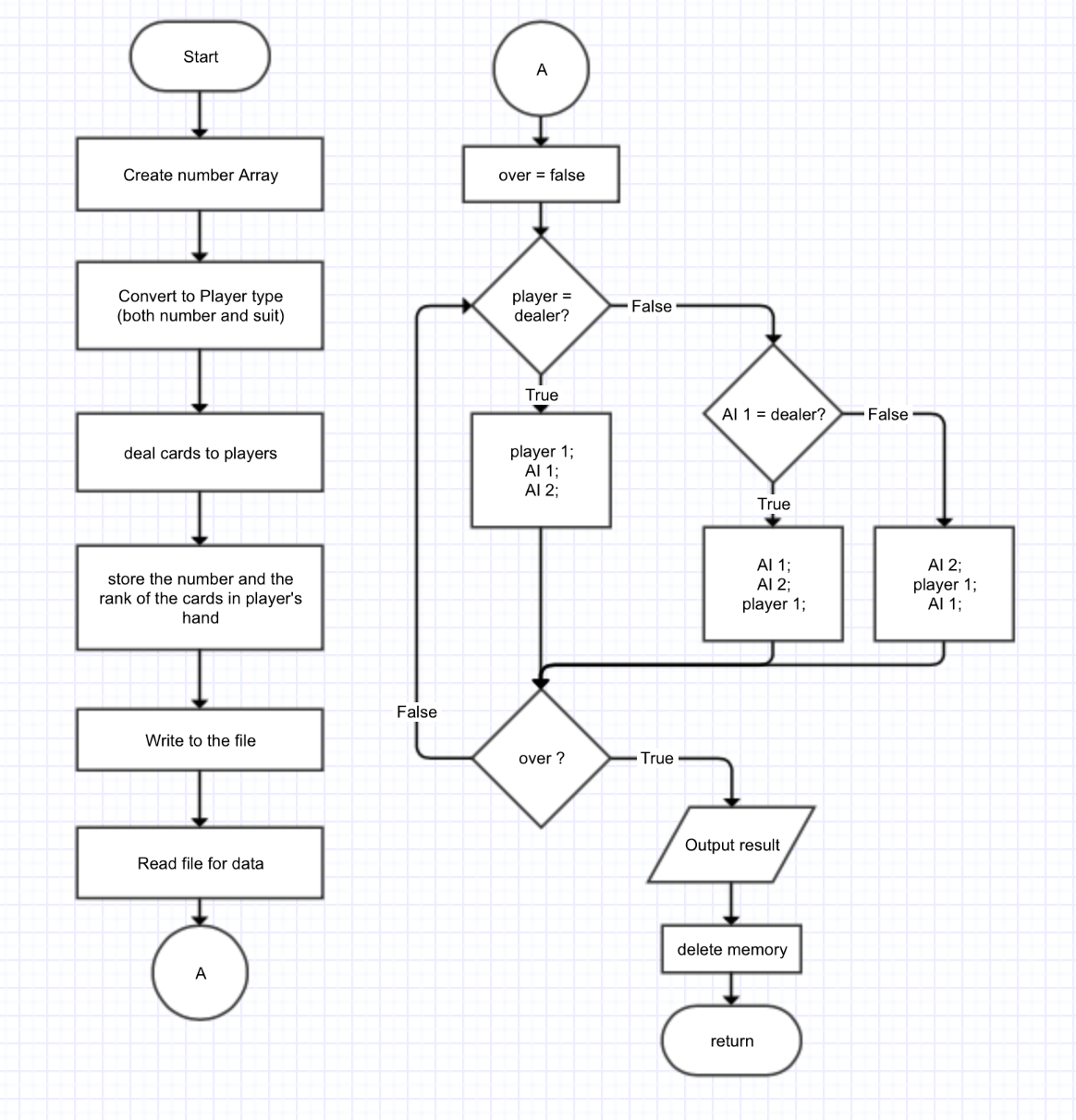


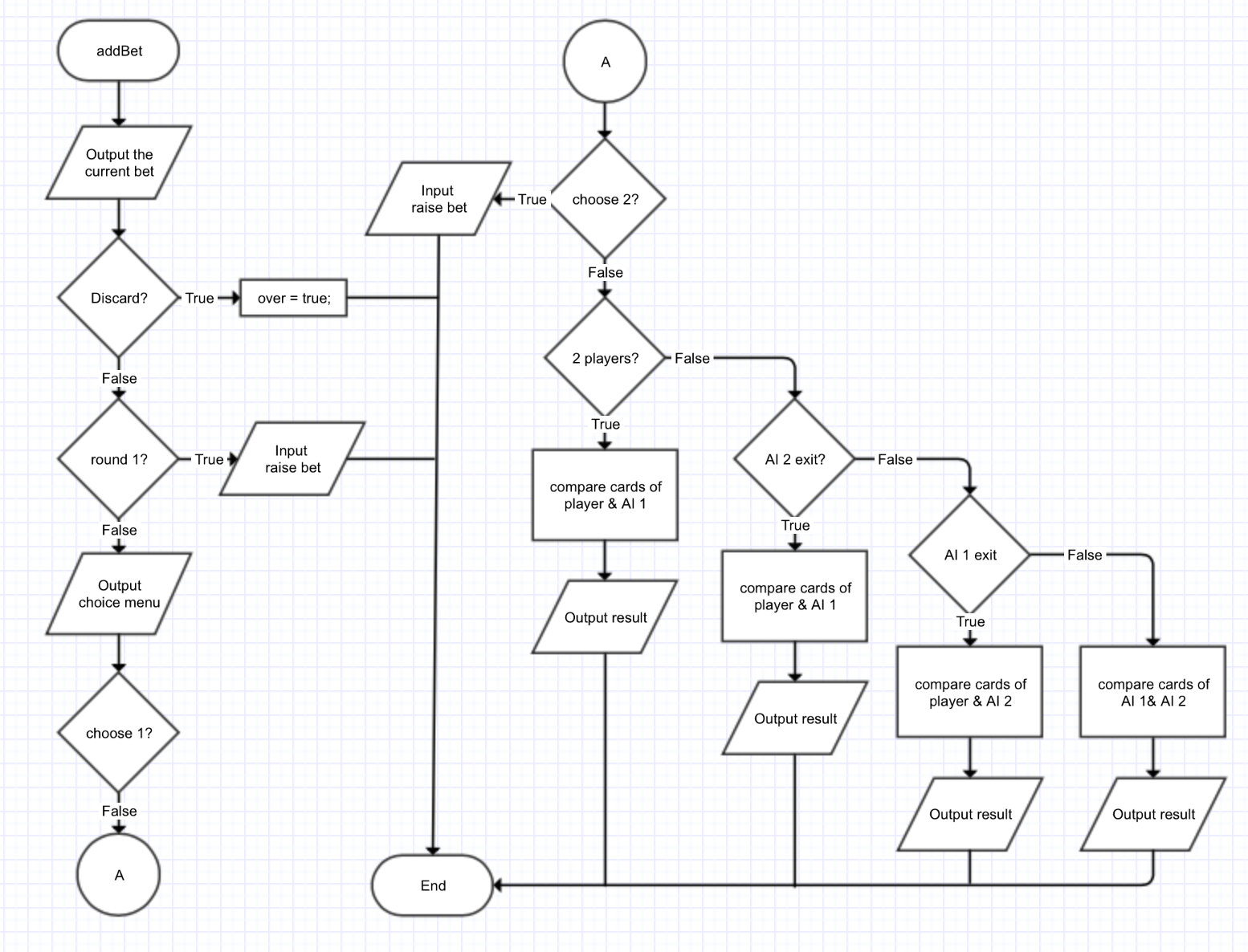
Choose 1 means to see

AI choose to challenge player

Accept challenge, and see the result

**B. Flowcharts**

Main:

addBet Function:

**C. Pseudocode**

* create a number array contains the number of 2~14
* convert the number array to Card array, containing suit and number
* promote the players number from player
* deal 3 cards to each player
* convert the 3 cards to number, and calculate the rank
* store all data about player into Player \*players
* write the data into the file
* read the data from file
* randomly choose a player as dealer
* If player:
  + ask if discard
  + during first round, promote for the raised bet
  + not first round, output choice menu
  + if 1: to see —— do nothing, calculate money
  + if 2 raise bet —— promote raised bet, calculate money
  + if 3 challenge ——
    - if 2 players —— compare player & AI 1
    - if 3 players ——
      * if AI 2 exit —— compare player & AI 1
      * else —— compare player & AI 2
* if AI:
  + AI I never discard
  + if 3 players —— AI 2 10% discard
  + if AI 2 not exit
  + if first round —— AI randomly raise bet
  + if not first —— randomly choose from 1~3 (1:2:2)
  + if choose 1 —— calculate money
  + if choose 2 —— randomly raise bet
  + if choose 3 ——
    - if AI 1:
      * if 2 players —— compare player & AI 1
      * if 3 players ——
        + if AI 2 exit, game over
        + else continue
    - If AI 2:
    - if AI 1 exit —— compare player & AI 2
    - else —— compare AI 1 & AI 2
* calculate the money of player
* output result (win or lose challenge, win or lose money)

**D. Variables**

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Variable Name | Description | Declare Location(line) |
| const int | Size | size of the number array | 33 |
| int | number[SIZE] | the number array of cards | 34 |
|  | swap | represent the suit | 34 |
|  | r | hold a random number | 108 |
|  | choice | the number of players | 126 |
|  | sortNum | temp hold number for sort | 164 |
|  | start | random choose dealer | 260 |
|  | max\_one | max bet in one round | 268 |
|  | max | max bet in the game | 269 |
|  | round | the round | 270 |
|  | bet | the bet on the table | 271 |
|  | money | the money player holds after game | 311 |
| bool | mak | sign to stop sort loop | 165 |
|  | over | game over | 263 |
|  | two\_exit | AI 1 exit game | 264 |
|  | thr\_exit | AI 2 exit game | 265 |
| Card | plyCard1 | Contain class pointer for player 1 | 151 |
|  | plyCard2 | Contain class pointer for player 2 | 269 |
|  | plyCard3 | Contain class pointer for player 3 | 270 |
| Player \* | players | class variable | 149 |

**E. Concepts**

|  |  |  |  |
| --- | --- | --- | --- |
| Concept | Type | Code | Location(line) |
| Binary file | fstream | ios::out | ios::binary  ios::in | ios::binary | 894  904 |
| Structure pointer | PlyNum | PlyNum \*plyNum; | 83 |
| Dynamic structure array | PlyNum\* | plyNum = new PlyNum[choice]; | 84 |
| Class pointer | Card | Card \*plyCard1; | 151 |
|  | Player | Player \*players; | 149 |
| Dynamic Class array | Player\* | players = new Player[3]; |  |
| Structure in a structure | Player | Card \* hold; | Player.h |
| Return structure from function | Card \*  Player \* | Card \*dealCard(Card \*, int);  Player \*playerInitial(Card \*, int); | 21  22 |
| Class as function argument | Player \* | void addBet(Player \*, bool&, bool&, bool&, int&, int&, int); | 23 |
| Constructor | Card | Card(); | Card.h  22 |
| Destructor | Player | ~Player() | Player.h  29 |
| Derive class | Public | class Player : public Card | 20 |
| Pointer in class | Card | Card \*card; | Player.h  22 |
| Return pointer in class | get function | Card \*getCard() const {  return card; } | Player.h  36 |
| const function in class | Int | int getNum() const | Card.h  25 |
| Bubble sort | int | sort = players[i].hold[j];  players[i].hold[j] = players[i].hold[j + 1];  players[i].hold[j + 1] = sort; | 162-183 |
| Switch | switch | switch(start) | 275 |
| Loop | for | for (int i = 0; i < SIZE; i++) | 79 |
|  | do-while | do {} while(mak) | 165 |
| Type casting | reinterpret\_cast  <type> | reinterpret\_cast<char \*> (p) | 896 |
| Random number | rand() | if (rand() % 5 < 3) | 621 |

**VI. UML**