Project 1

<Liar Dice>

<Version 2.0>

Name: Haolan Ye (Benjamin)

Class: CSC-17A 43950

Date: 06/04/2015

**Contents**

1. **Introduction……………………………………...2**
2. **Summary……………………………………........3**
3. **Problems during coding…………………………3**
4. **Pseudo Code…………………………………...…5**
5. **Screen shot…………..…………..…………..........6**
6. **System Libraries……………………………...…10**
7. **Concept Covered…..…….……..……..………....11**
8. **Flowchart...…………..…………………………..13**
9. **UML...………………..…………………………..20**
10. **Introduction(** <http://en.wikipedia.org/wiki/Liar's_dice> **):**

Five dice are used per player with dice cups used for concealment. Each round, each player rolls a "hand" of dice under their cup and looks at their hand while keeping it concealed from the other players.

The first player begins bidding, announcing any face value and the number of dice that the player believes are showing that value, under all of the cups in the game. Ones are often wild, always counting as the face of the current bid. Each player has two choices during their turn: to make a higher bid, or challenge the previous bid - typically with a call of "liar." Raising the bid means either increasing the quantity, or the face value, or both.

If the current player challenges the previous bid, all dice are revealed. If the bid is valid (at least as many of the face value and any wild aces are showing as were bid), the bidder wins. Otherwise, the challenger wins.

That game I made was created based on the way I played in China. Because of difference between different regions, some rules of Liar Dice are different. In China, there are some addition rules (also in my game):

* From the beginning, ones are wild unless you bid 3 ones
* After bidding ones, ones cannot be wild anymore
* You could bid only 3 fives, which means you bid there are 3 fives but not including ones
* After bidding only 3 fives, ones cannot be wild anymore
* The number of face of first bid has to be greater than 1.5\*players

1. **Summary:**

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

This game contains most concepts that we have learned in the class. I used pointer with player (structure) and used structure to record the dices that each player has. In the structure of player, there is also a tag (integer) for the player. I will use the tag when someone wants to challenge. The game will write the data of players into binary file, and after someone challenge, it read the file to a new players array. Afterward, it will get the result by using the new players array.

1. **Problems during coding**
2. **Limit the player input with correct format**

When player doesn’t challenge, he need to make a higher bid. Player needs to input a string for bidding. “4 5” means that player bids 4 fives. “4n5” means that player bids 4 fives only (ones cannot be wild at that time).

1. **Get the playing order for the players**

At the beginning of the game, it will randomly get the playing order for the players. In the rest of the game, players bid and challenge based on that order. I used a switch statement in a do-while statement to randomly access. It loops until someone challenges.

1. **What should AI do?**

There is no a specific algorithm for the AI in that game. I made the AI based on what I think when I play Liar Dice. There are lots of possibilities that happens when AI determine challenging or not.

* When AI doesn’t have the dices that bided by previous players
* When AI only have one that bided by previous players

… etc.

When AI needs to bid higher, AI should sometimes lie and sometimes tell the truth. Therefore, I set the possibility that AI lie to 2/5. When AI tells the truth, he will bid based on what dices he has. When AI lies, he will randomly select one face of dice that does not exist in his dices.

1. **One is wild**

Mostly, ones are wild unless you bid “3 fives only” or “3 ones” (Both are example). Therefore, I need a Boolean to record one is wild or not. After one is not wild, the number of each dices doesn’t count ones.

1. **Pseudo Code**

Set seed for random number

Sign in/sign up (input name, password, email)

Introduce the game

Display menu (play, add coins, or exit)

When Play, Prompt players for the number of players

Roll the dices for players

Initialize based on the number of player

Display player’s dices

Randomly choose a player be the first bidder

Begin biding until someone challenge {

Someone bid first (based on random select)

Other players determine challenge or not

Bid in players order

}

Show dices of all players

Display the result of the game {

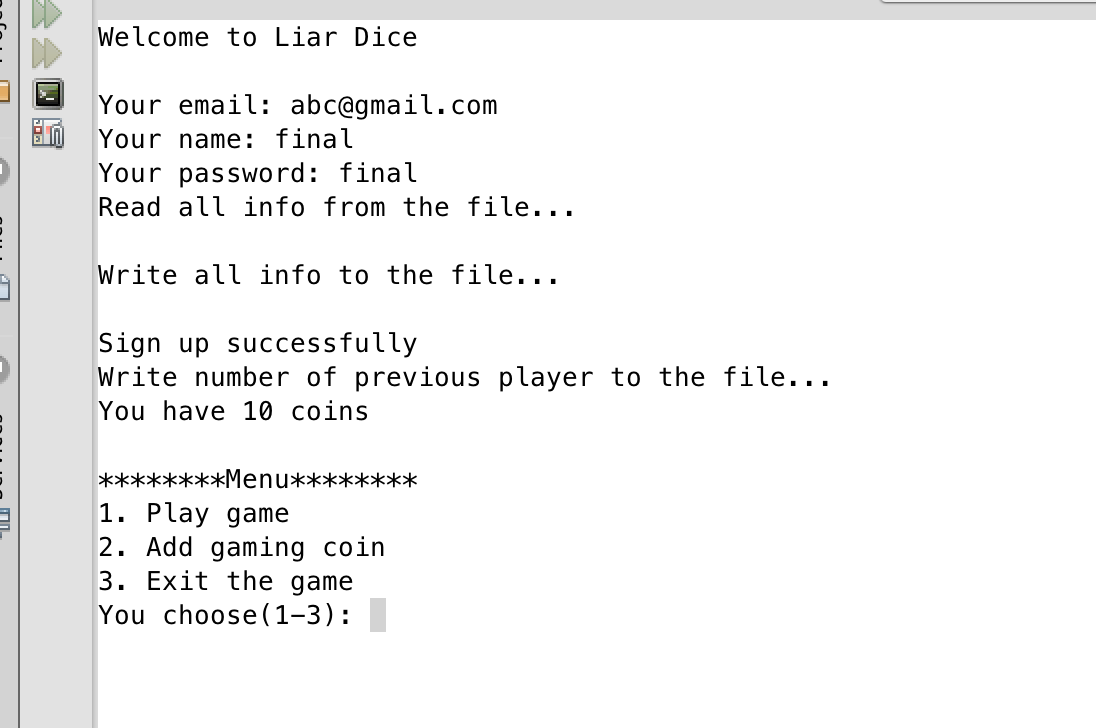
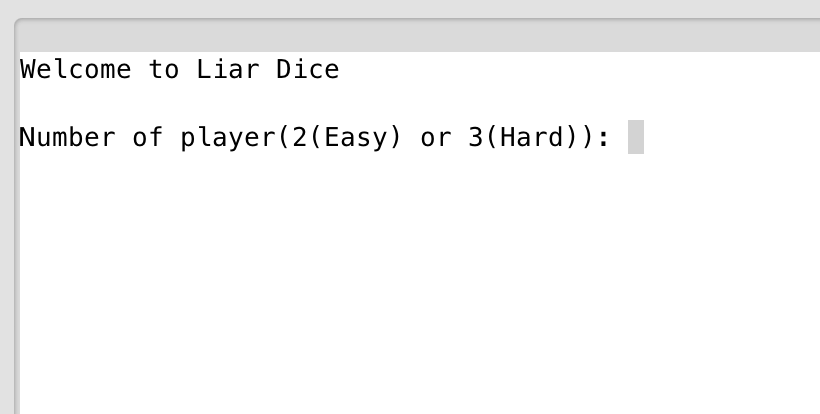
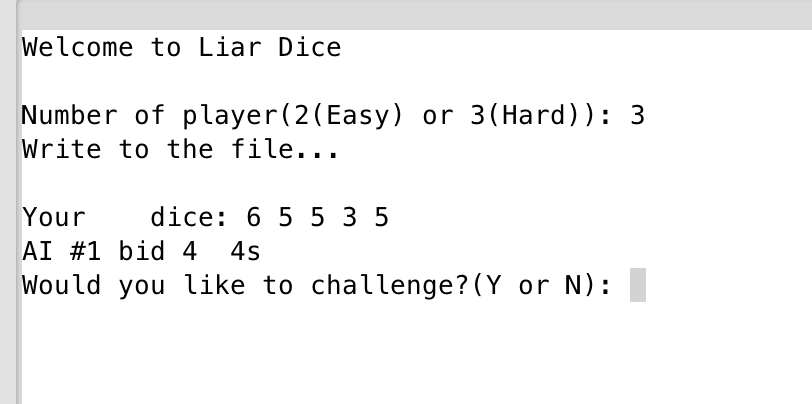
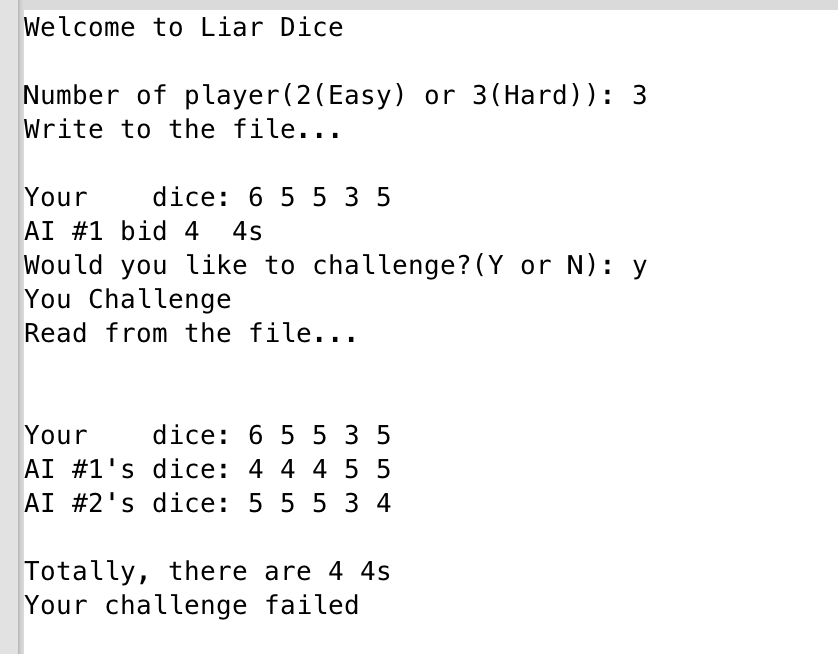
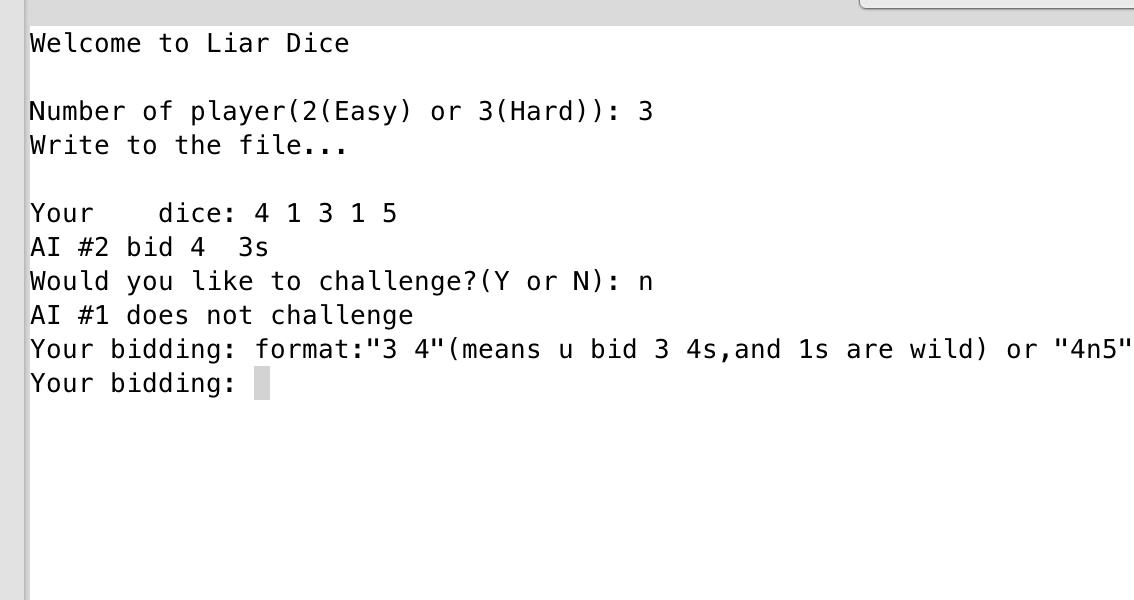
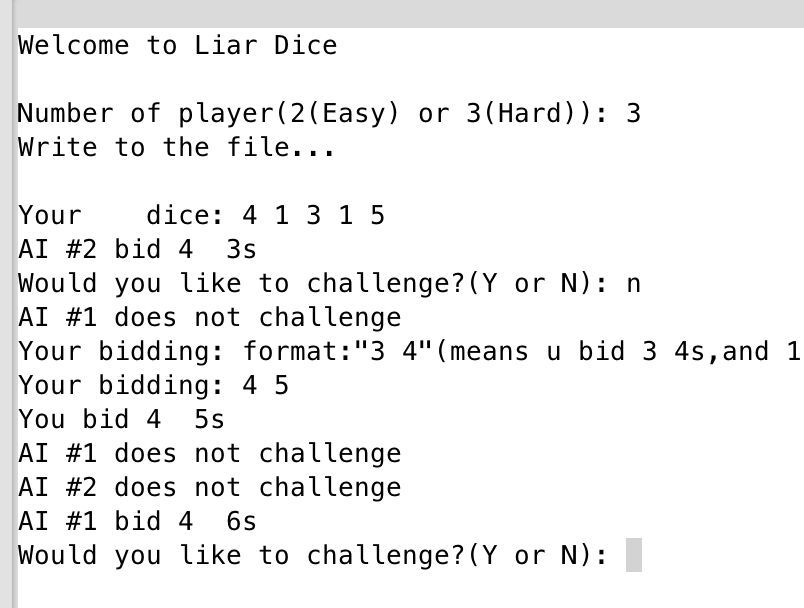
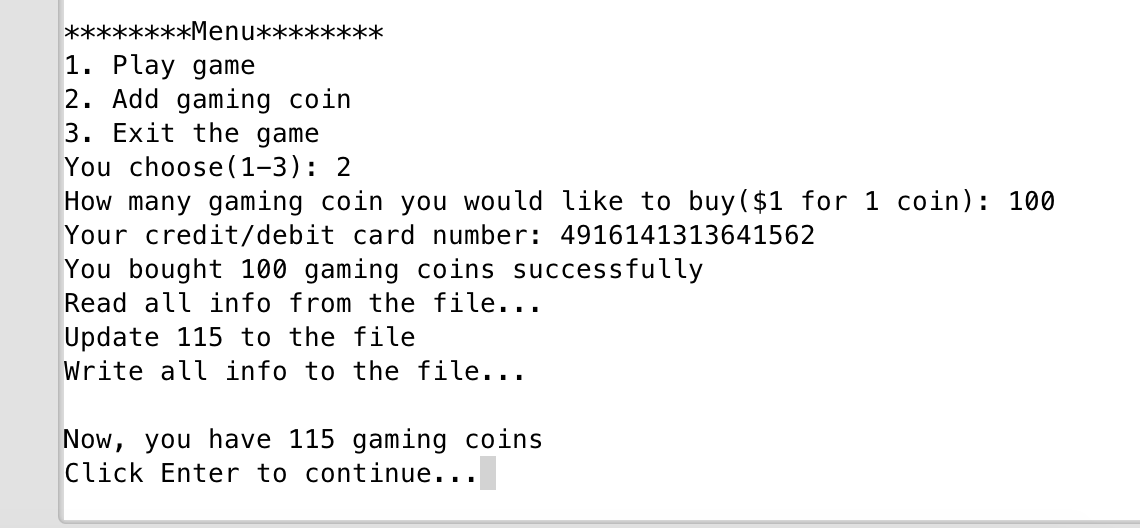
Check the number of face bided is greater than the number of dices that players have exactly

}

update the information of players with binary file

When add coins, prompt user input the card number

Deallocate memory

1. **Screen Shot**
2. Ask for number of player
3. Ask for number of player 
4. Randomly choose a player to be the first bid, then you can challenge
5. If challenge, the result will come up
6. If not challenge, your turn to bid
7. After you bid, if AI(s) doesn’t challenge, it’s their turn to bid, and ask you challenge or not 
8. Add gaming coins
9. **Libraries**

**a. System libraries**

* **<iostream>**
* **<cstdlib>**
* **<ctime>**
* **<string>**
* **<vector>**
* **<fstream>**

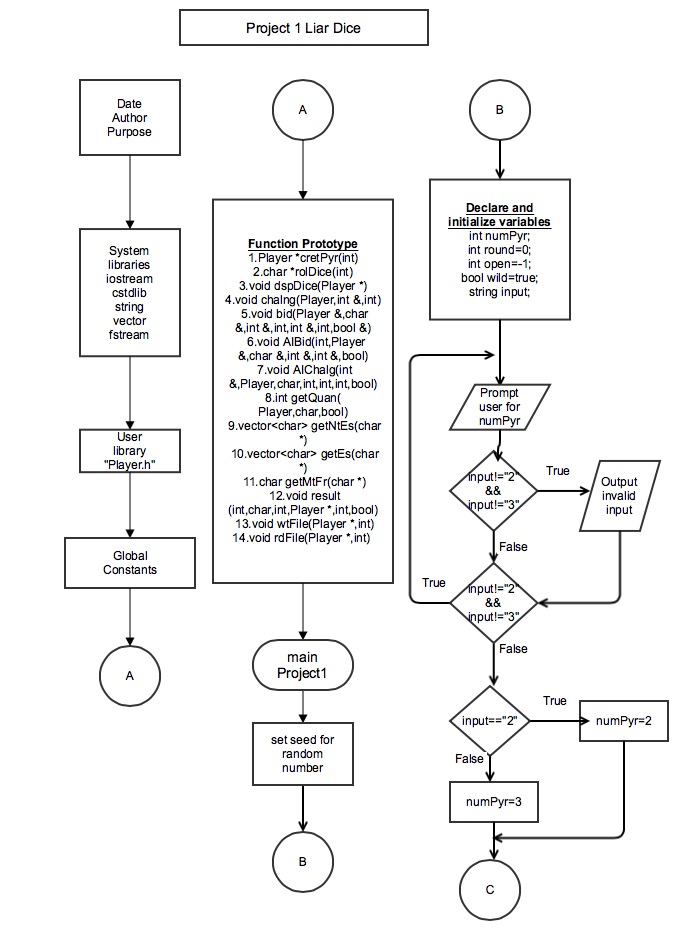
**b. System libraries**

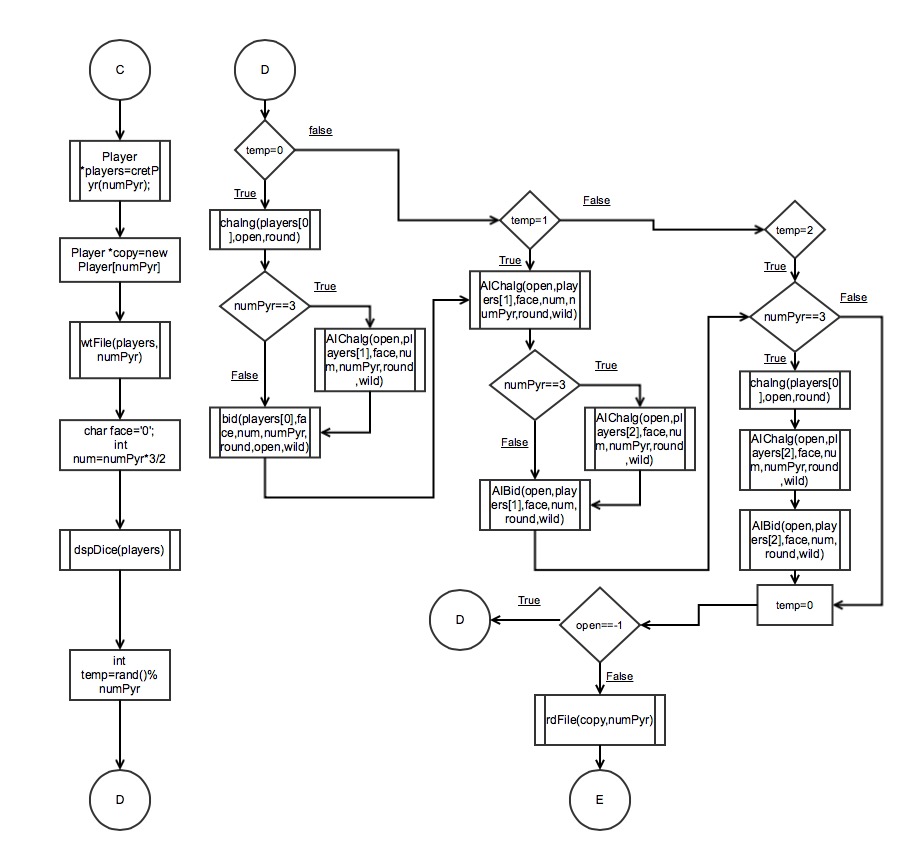
* **AI.h**
* **Player.h**

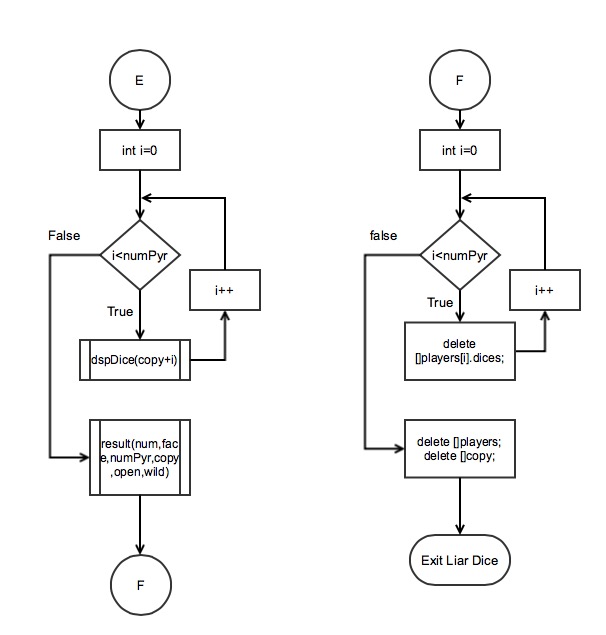
1. **Concept covered**

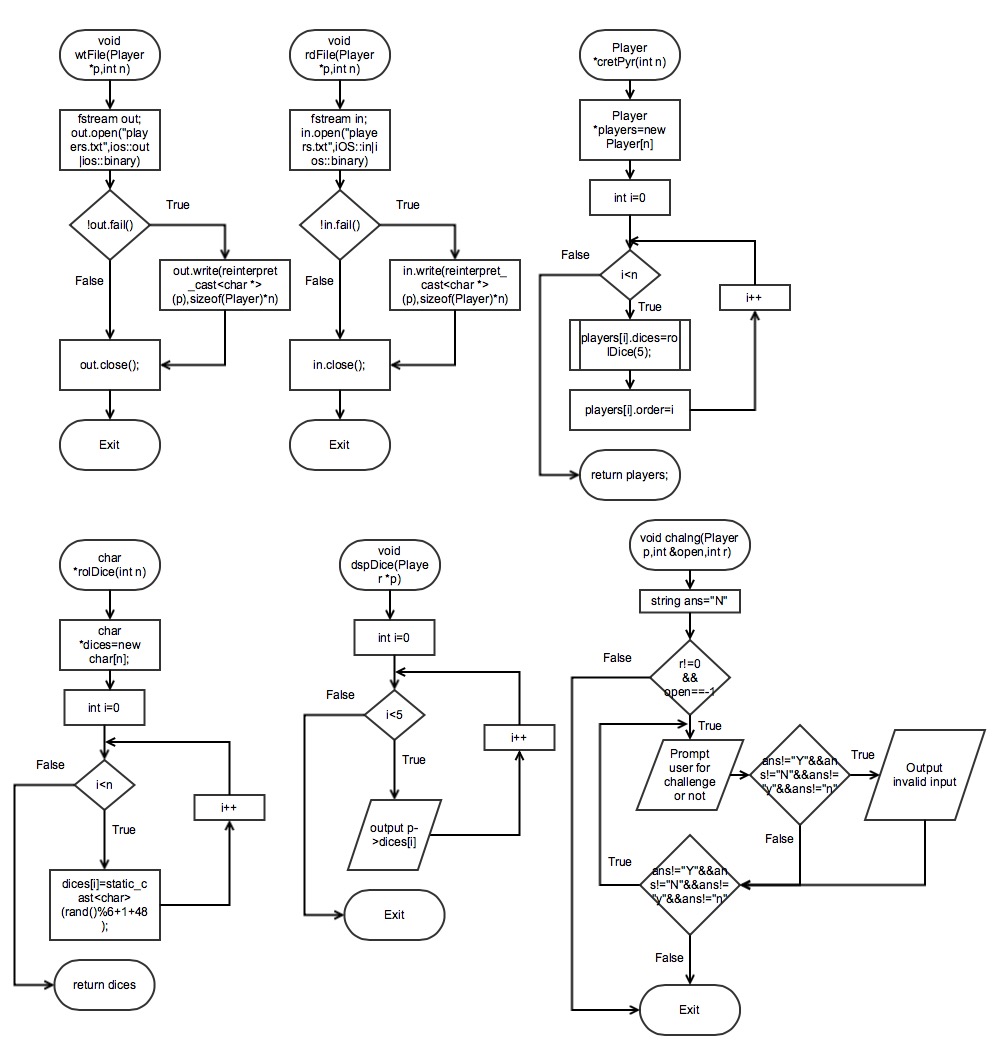
|  |  |  |  |
| --- | --- | --- | --- |
| Concept | Type | Code | Location(line) |
| Pointer with structure | Info \* | Info \*infor=new Info[getNInf()]; | 259 in player.cpp |
| Type casting | static\_cast<type> | static\_cast<unsigned short>(time(0)) | 35 in main |
| Binary file | fstream output | fstream out; | 274 in player.cpp |
|  | fstream input | fstream in; | 260 in player.cpp |
| string | string | string ans="N" | 161 |
| class | class | class Player | 14 in player.h |
| getter | get | int getOrdr() const {return order;} | 64 in player.h |
| setter | set | void setNInf(int); | 33 in player.h |
| inline function |  | void setOrdr(int n) {order=n;}; | 51 player.h |
| Constructor |  | Player::Player() | 29 player.cpp |
| Overload function |  | void bid() | 25 in AI.h |
| abstract base class |  | virtual void bid()=0; | AbsPlayer.h |

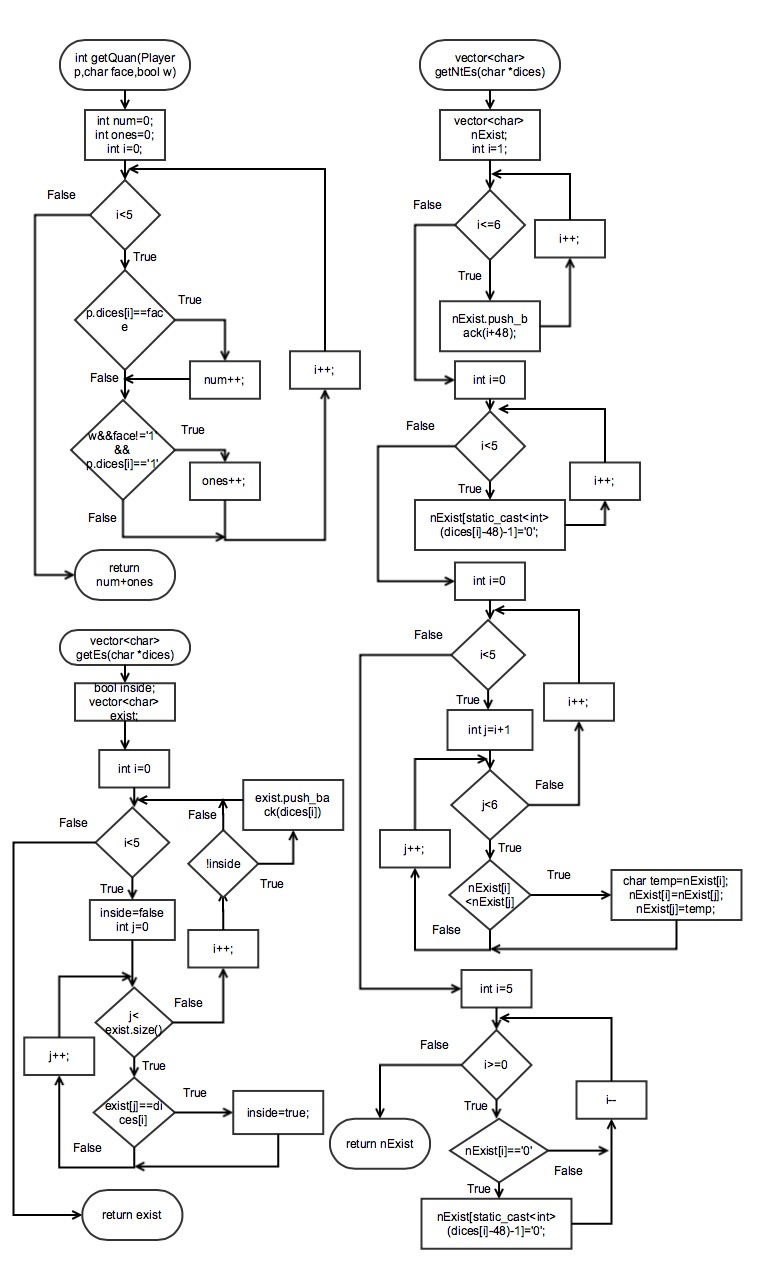
|  |  |  |  |
| --- | --- | --- | --- |
| vector | vector<char> | vector<char> nExist | 15 AI.h |
| Sorting | Sorting | for(int i=0;i<5;i++) {  for(int j=i+1;j<6;j++) {  if(nExist[i]<nExist[j]) {  char temp=nExist[i];  nExist[i]=nExist[j];  nExist[j]=temp;  }  }  } | 332 |
| Array of Object |  | AI \*a=new AI[np-1]; | 60 in main |
| Instance variables | static int | static int open | 21 in Player.h |
| Static Member Functions |  | static void setNumC(); | 23 in Player.h |
| Operator Overloading |  | T &operator[](const int &); | 32 in aVector.h |
| inheritance |  | class AI:public Player | 12 in Player.cpp |
| Template class |  | template <class T>  class aVector | 16 in aVector.h |
| Exception |  | try {  aptr = new T[usdSize];  } catch (bad\_alloc) {  memError();  } | 45 in aVector.h |

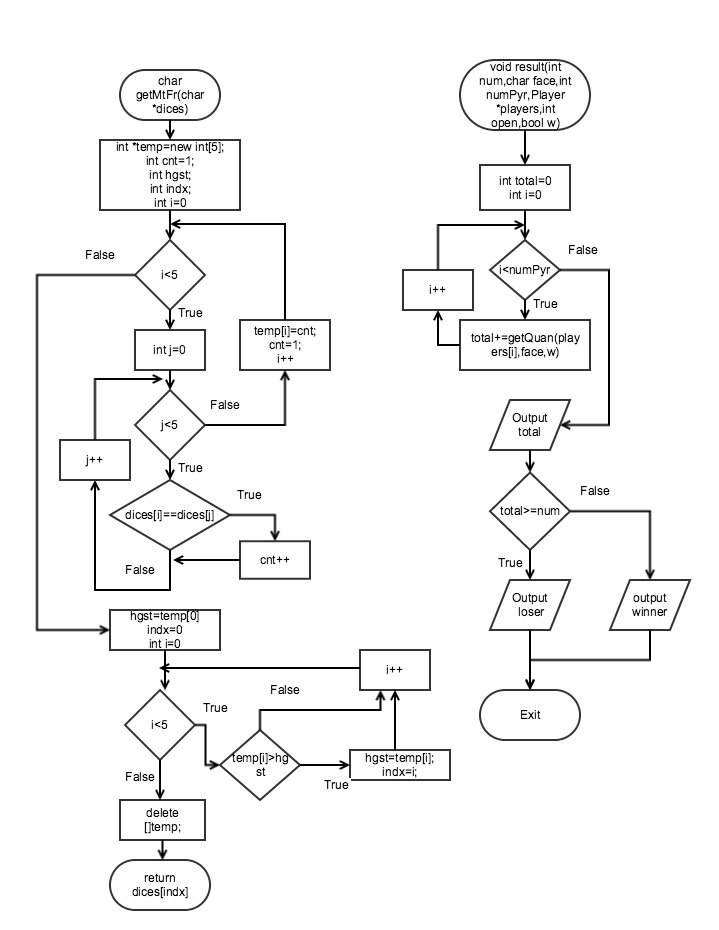
1. **Flowchart in Project 1 (1) Main flowchart (3 pages)**

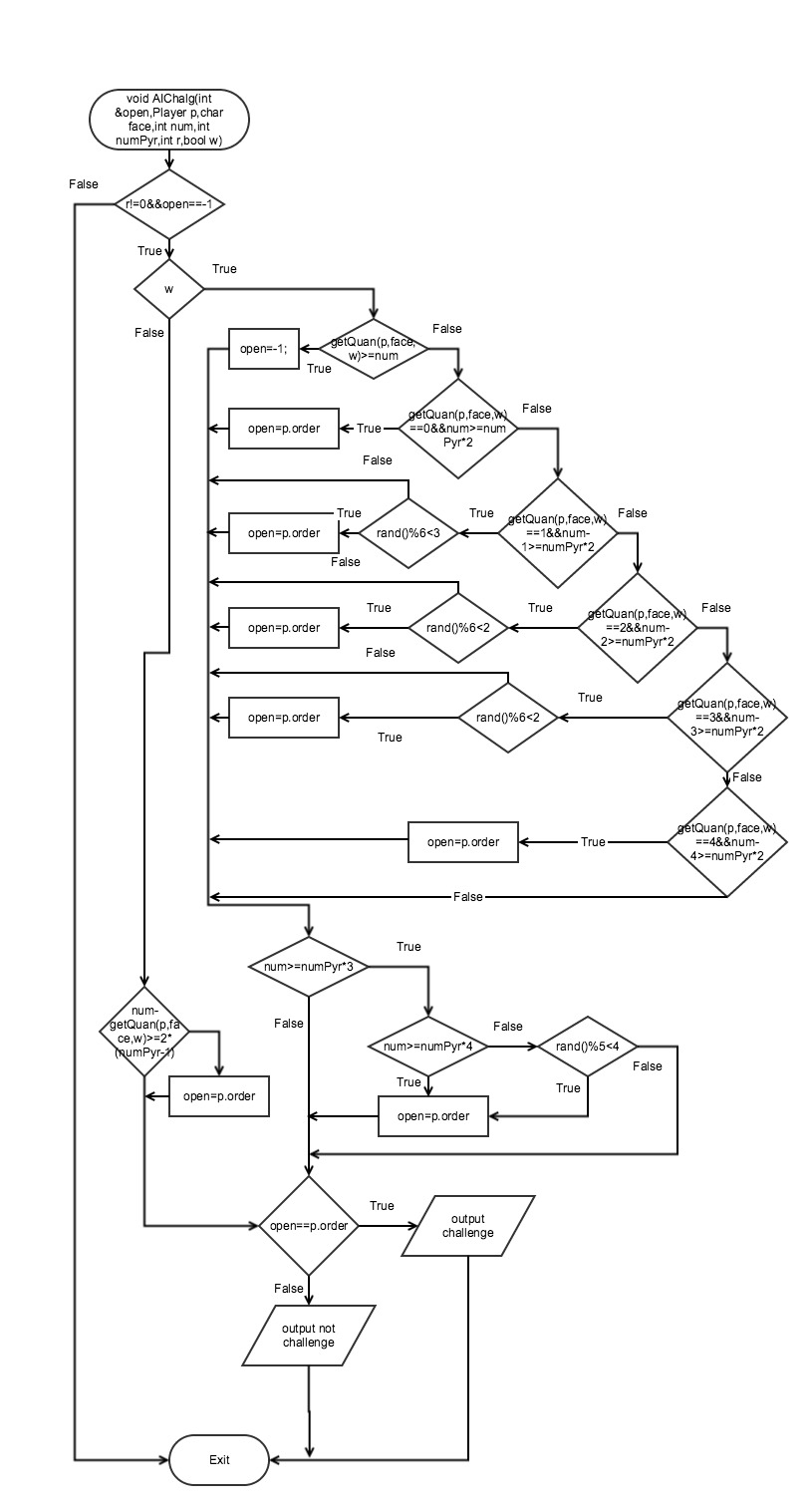




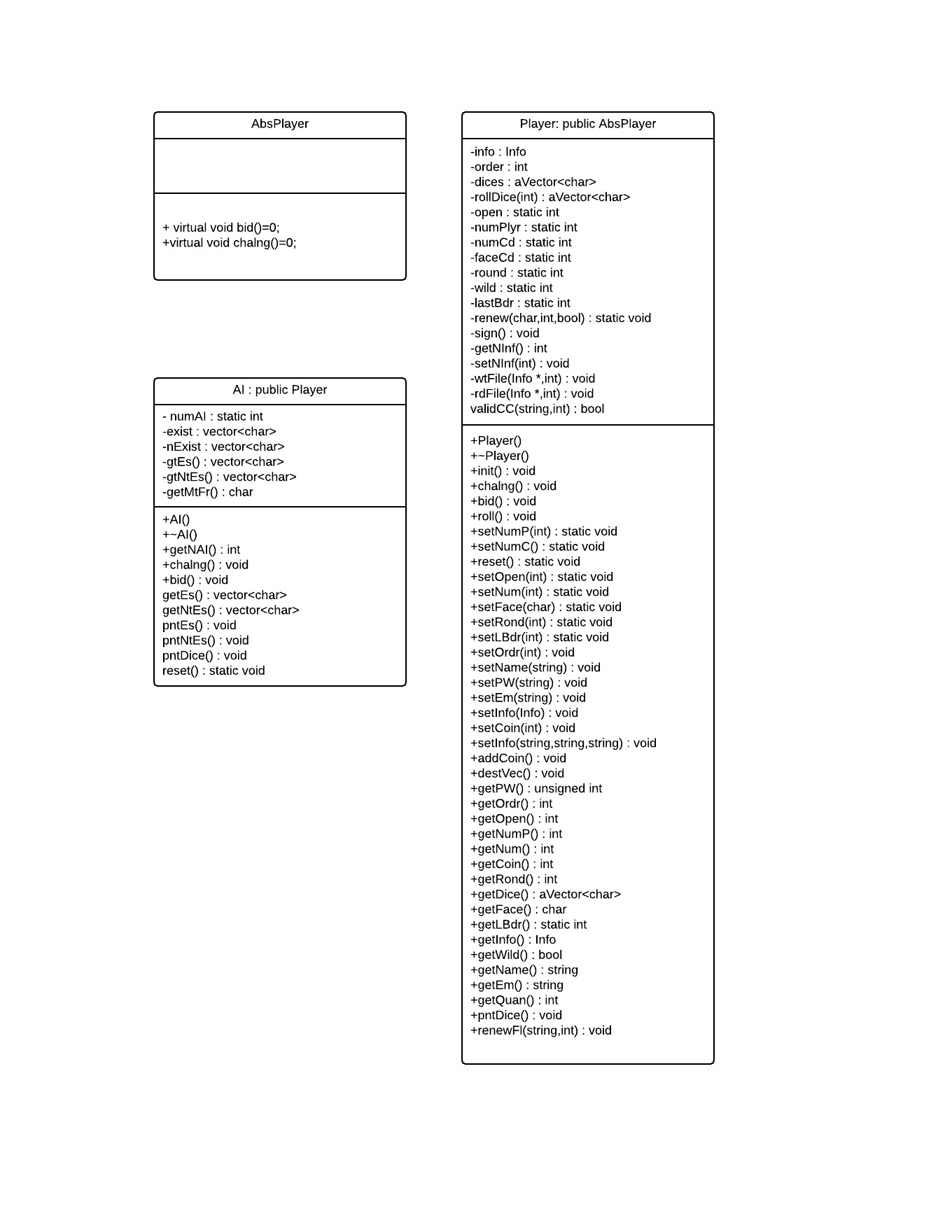
**(2) Function flowchart (3 pages)**







1. **UML**

****