Project 2

Blackjacks

Louidor, Richemond

Course: CSC 5

SEC: 46091

DATE: 08/2/15

Table of Contents

Topics:

1. **Introduction**
2. **Game Play and Rules**
3. **Development and Logic**
4. **Highlights**
5. **Reference**
6. **Highlights**
7. **Summary**

INTRODUCTION:

Blackjack, also known as twenty-one it is thought that it origins started in France back in the 1700s. Blackjack is widely known and popular game in the United States, mostly

common in Casinos. Nowadays, blackjack can be played right from home or on the go either on a smartphone or on a personal computer. Finally I get my chance of creating my own version of blackjack.

Game Rule:

1. Winning
   a. Player(s) hit Blackjack (21) on first deal.
   b. Player sum is greater than dealer sum after additional cards are dealt.
   c. Player(s) hit Blackjack (21) after additional cards are dealt.
   d. Dealer bust (dealer sum is >21)
2. Losing
   a. Dealer hit Blackjack.
   b. Player(s) Bust during additional card dealing
   c. Player(s) sum less than dealer card sums.
   d. dealer cheats.
3. Push
   a. Player(s) sum equal dealer sum (pSum[] == dSum)

```
loui total: 15
hit(1) or stay(2): 2

rich total: 5
hit(1) or stay(2): 1

New Card: 9  new total: 14
enter 1 for hit or 2 for stay 1
New Card: 1  new total: 15
enter 1 for hit or 2 for stay 2
hanes total: 20
hit(1) or stay(2): 2

loui Total: 15
rich Total: 15
hanes Total: 20
```

Game Play:

1. Enter number of players.
   a. Players input name
   b. Players input chips buy in
   c. Players place bets

2. Dealer First two cards are dealt.
   a. Dealer card 1 is reveal
   b. Dealer card 2 is hidden

3. Players First two cards are dealt.
   a. Players cards are added up making sure player do not bust on first dealt or if player hit blackjack for an automatic win.
   b. If player(s) sum is less than 21 player decides to hit or stay.
   c. Hit ->to deal a 3rd card and beyond or stay with current cards.
4. Hit
   a. Player(s) sum is greater than 21, player(s) lose.
   b. Player(s) sum is smaller than dealer player(s) lose
   c. Player(s) sum is equal to 21, player wins .
   d. Player(s) sum is greater than dealer, players win

5. Stay -> stay with current cards.
6. Dealer:
   a. Checking dealer sum to make sure dealer doesn't bust on first dealt.
   b. Dealer sum is equal to 21 player(s) lose
   c. Dealer hit if sum of first two cards are less than 17.

d. Dealer stay when sum is equal to or greater than 17.
e. Player(s) win if dealer sum is greater than 21 during dealer hit.

7. Dealer sum and Player(s) sum are being compare the greater value wins.
8. Game prompt player(s) to play again y/n.
9. Yes: game starts over again, No: game ends, thank you for playing message displays.

10. Players name, remaining chips and last place bets are stored in “BJK.txt” file.

DEVELOPMENT AND LOGIC

System Libraries used: Line(10-15)

- Iostream -> C++ library deals with namespace std
- cstdlib -> C++ standard library
- fstream -> file i/o, write to file and read from file
- ctime -> ctime library helps generate the random values.
- Vector -> Vector library
- String -> string library

User libraries: line(23-28)

- Struct pData line(23-28, 58-61, 228-236):
    1. name
    2. bet
    3. chip

Function Prototypes: line(33-51)

- Error1-> check for player first hand bust, blackjack, two aces drawn
- Error2 -> check for bets, and chips error
- Error3 -> check for aces during player hit's
- dDeal -> dealer deal 3rd card
- dError1-> check for dealer blackjack, bust, and two aces drawn after second card
- dError2 -> check for dealer hit for aces drawn
- hitStay -> Player chooses to hit or stay
- dDeal -> Dealer cards are reveal and third card is dealt
- win-> Player(s) wins
- lose -> Player(s) loses
- pJack -> Player(s) or dealer hit blackjacks
- pBust -> Player(s) bust sum is over 21

- dBust -> Dealer bust sum is over 21
- push -> Game push dealer sum equals player sum
- rFile-> reading from file
- srt1 -> sort chips ascending order
- srt2 -> sort bets descending order

Dynamic Arrays line(56-58)

- Name
- Chips
- Bets

Variables Declared: line(62-77)

- card1[] ->Players card 2
- card2[] ->Players card 2
- dCard1 -> Dealer card 1
- dCard2 -> Dealer card 2
- dSum -> Dealer sum
- SIZE-> number of players
- pSum[] ->Players sum
- count -> Game counter
- name []-> Players name
- chips[] -> user chips
- bets[] -> user bets

2. user prompt to enter number of players
   - input SIZE
3. while (SIZE < 0) prompt use to enter a number greater than zero.
4. Getting player Names and Chips. Also error check for negative numbers for chips
5. User place bets
6. Dealer card 1 and card 2: hidden not reveal
7. Card error checks
   - dError1
   - dSum=22?

- dCards 1&1?
- dCards 10& 1?
- dCards 11&1?

8. players card are dealt
    - error1
    - pSum=22?
    - Cards 1&1?
    - Cards 11&1?
    - Cards 10&1?

9. players hit or stay
    - hitStay
    - card 3
    - error2
        1. sum<11 & card 1?
        2. Sum>=11 & card =11?

```
363 /*****************************************************************************
364  ********************************** error2 ***********************************
365  * Purpose: checks for user third card draw for aces
366  * (sum > 11 && card == 11) -> sum +=1
367  *
368  *  (sum[x] < 11 && card == 1) -> sum+=11
369  */
370 void error2(int card, int sum[], int SZ)
371 {
372     for (int x = 0; x < SZ; x++)
373     {
374         if (sum[x] > 11 && card == 11)
375         {
376             sum[x] += 1;
377         }
378         else if (sum[x] < 11 && card == 1)
379         {
380             sum[x] += 11;
381         }
382     }
383 }
```

10. dDeal
    - Dealer sum less than 17 dealer deals third card

- If dealer sum is greater or equal to seventeen and less than 21 then dealer sum get pass back to main.

11. Win, lose, push, dBjk, pJack, dBust, pBust

```
void win(string name[], int chips[], int bets[], int pSum[], int dSum, int SZ)
{
    for (int x=0; x<SZ; x++)
    {
        if((pSum[x]>dSum && pSum[x]<21))
        {
            cout <<"YOU WIN\n";
            chips[x]+=bets[x];
            cout <<name[x]<<" chips total: $"<<chips[x]<<endl;

        }
       }
     cout<<endl<<endl;

}
/*****************************************************************************
 ********************************* pJack *********************************
 * purpose: Player hits blackjack when ((pSum==21 && sSum !=21))
 */
void pJack(string name[], int chips[], int bets[], int pSum[], int dSum, int SZ)
{
    for (int x = 0; x < SZ; x++)
    {
        if (pSum[x] == 21 && dSum!=21)
        {
           chips[x]+=bets[x];
           cout << "BLACKJACK\n"<<endl;
           cout <<name[x]<<" chips total: $"<<chips[x]<<endl;
        }

    }
```

```
586 void pBust(string name[], int chips[], int bets[], int pSum[], int dSum, int SZ)
587 {
588     for (int x = 0; x < SZ; x++)
589     {
590         if (pSum[x] > 21)
591         {
592             cout << "BUST!!! YOU LOSE\n";
593             chips[x]-=bets[x];
594             cout <<name[x]<<" chips total: $"<<chips[x]<<endl;
595             cout<<endl;
596         }
597     }
598
599
600 }
601 /*****************************************************************************
602  ********************************* push *********************************
603  * purpose: Player push when ((pSum == dSum))
604  */
605 void push(string name[], int chips[], int bets[], int pSum[], int dSum, int SZ)
606 {
607     for (int x=0; x<SZ; x++)
608         {
609         if(pSum[x]==dSum && dSum!=21)
610         {
611
612            cout <<"Push\n";
613            cout <<name[x]<<" chips total: $"<<chips[x]<<endl;
614             cout<<endl;
615         }
616     }
617
618 }
```

12. Player is prompt to play again or to quit.
    - Play again process starts all over.
    - Quit thank message is displayed
    - Game ends
    - Players name, remaining chips and last best are written to “struct.txt” files
    - Srt1()
        1. Sorts bets from largest to smallest
        2. Vector usage is implemented
    - Srt2()
        1. Sorts chips from smallest to largest
        2. Vector usage is implemented
        3. Vector numbers from smallest to largest

```
749     for (unsigned int i = 0; i < SZ; i++)
750     {
751         cout <<"bets "<<bets[i]<<endl<<endl;
752         vBets.reserve(vBets.size() + SZ);
753         vBets.push_back(bets[i]);
754         copy(&bets[i], &bets[SZ], back_inserter(vBets));
755 
756         //result output to both file and console output
757         file << "$" << vBets[i] << endl;
758     }
759     file.close();
```

```
793     for (unsigned int i = 0; i < SZ; i++)
794     {
795 
796         vChips.reserve(vChips.size() + SZ);
797         vChips.push_back(chips[i]);
            copy(&chips[i], &chips[SZ], back_inserter(vChips));
799 
800         //result output to both file and console output
801         file << "$" << vChips[i] << endl;
802         cout << "$" << vChips[i] << endl;
803     }
804     file.close();
```

HIGHLIGHTS

Struct pData usage (line(23-28, 58-61, 228-236))

```
58      //struct
59      pData* Name= new pData[50];
60      pData* Chips = new pData[50];
61      pData* Bets = new pData[50];
```

```
227     for (int x = 0; x < SIZE; x++)
228     {
229         Name[x].NAME = name[x];
230         file << "\n" << Name[x].NAME << endl;
231
232         Chips[x].chip = chips[x];
233         file << "Chips " << ": $" << Chips[x].chip << endl;
234         Bets[x].chip = bets[x];
235         file << "Bets " << ": $" << Bets[x].chip << endl;
236     }
```

Dynamic Array usage(line(59-61, 238-240))

```
58      //struct
59      pData* Name= new pData[50];
60      pData* Chips = new pData[50];
61      pData* Bets = new pData[50];
```

```
237     //delete dynamic arrays
238     delete [] Name;
239     delete [] Chips;
240     delete [] Bets;
```

Vectors usage srt1() & srt2() lines(710-806)

```
769     ofstream file;
770     file.open("vChips.txt");
771     vector<int> vChips(0);
772
```

```
711 {
712     ofstream file;
713     file.open("vBets.txt");
714     vector<int> vBets;
715     for (int i = 0; i < SZ; i++)
716     {
```

Read From Files

```
812         ifstream file;
813         string bets;
814         file.open("vBets.txt");
815         cout << "largest to smallest\nLast Placed ";
816         while (getline(file, bets))
817         {
818             file>>bets;
819             cout << bets << "\n";
820         }
821         file.close();
```

Summary

Blackjack 2, has many different concepts added on to it. The game was rebuilt from ground up to make it easier in implementing new concepts and logics. Before, Blackjack was one player and was consists of mainly if and else statements. In blackjack 2 new concepts such as parallel arrays and dynamic arrays were implemented to store user data. Therefore giving the program the ability to handle multiplayers at a time. There are no set limits on the size of players that can play blackjack 2 at a time. Player data were also stored in structure data type name pData. It stores users name, remaining chips and last bet place and exported it to a .txt file for records. Functions were implemented as a way to make the code look neater and easier to debug. Approximately about 20 functions were created to handle the logic behind blackjack 2. Sorting function was created to sort users bet from highest to lowest. Reading from file, vector implementations, passing arrays to functions, error checking players individually were all new logics that was implemented to get blackjack 2 up and running. Furthermore there are lots more potentials that can be done with developing blackjack 2 into 3. For instance, developing a bank account for dealer and store the data in a separate private file. When players purchase credits for chips from dealers those credits will get be added to dealers bank account. Also, give the user the ability to exit the game if they wish for. Blackjack 2 was a great final project because of how tedious it was and paying very close attention to smallest of details.

Reference

1. http://www.stackoverflow.com
2. http://www.cplusplus.com/
3. Textbook -> Tony Gadis 7th Ed.
4. Youtube.com C++ tutorial videos