Project 2

<Blackjack>

Izeth Torres
CSC-CIS-5
Spring 2018
42375

Table of contents

Introduction

Game: Blackjack

Blackjack is a card game in which a player plays against the house. The goal of this game is to get your cards to add up as close to 21 as possible without going it, while at the same time scoring higher than the house.

This game consists of a deck of 52 playing cards. In which each card with a number has a value of that number. As for the cards with letter, they have specific values. The Jack, Queen and King cards are all worth 10 points each. The Ace card can either be worth 11 or 1, depending on the players choice and which value suits their game best.

Both the player and the house begin with two cards each. The player can see their cards but can only see one of the house cards. The player is then asked is than asked if the want a new card in addition to the tow they have. The player can decide to receive as many cards as they want as long as the cards total does not go over 21. If the player goes over the maximum of 21 then the player automatically loose.

The house will keep asking for cards as long as their total is 17 or less. However many cards the house excepts is unknown to the player but the first face up card be seen throughout the game.

If you are betting in this game then you would place the minimum bet at the beginning and then be able to add to the pot once you have chosen you maximum amount of cards.

Summary

Project size: Approximately 260 lines

This version of blackjack is one player vs. the house. The player will begin the game and on the screen they will see their two cards with the total of those cards as well as one of the house's cards. The player will then be given the option to request a new card or keep the cards they have.

Every time the player decides they want a new card, the new card will appear as well as the total of the cards after every card they draw.
When the player decides they do not want any more cards, the game will end displaying the results of the game and the value of the house's cards. The game will either display a winning message, losing message or a tie message.

Translating game to programing

In order to generate a deck of cards I would have to create a program dedicated to the specific task of generating and shuffling a deck of cards. I would then have to figure out how to use that program in the game I created. Since we have not learned how to do that in class yet, I went forwards with a way of mimicking a deck of cards. This is where functions became useful. I was able to create a function that gives values to the cards, as well as create suits for them.

The task with creating a deck was accomplished, but how would I go about distributing the cards to the player and house? Both player and house must begin with 2 cards each so I created an array that holds the first two cards and set them equal to a function that calls for the next card in the deck.

Functions were very useful in breaking up the code to perform specific tasks and well as make it look more organized. Within the functions I was able to create arrays that would shuffle the deck, distribute the cards as well as print the cards and tally up their total.

When deciding to initialize the game I thought about placing it in a void function but then thought it would probably be better to see the game play out in the main function and then call for other functions. The layout of the game begins with setting up the cards for the house and the player. It then moves on to displaying the first hand and requesting a response from the player on whether they want a new card or not. The program then keeps tally of the cards as the player requests more until the player decides they don't want any cards or the player goes over 21. Once the game is over the program will close.

The thought of adding a betting function did cross my mind. I would have had to create a new function to call for a wager and then tally up that amount in the main function. But being overwhelmed with amount of functions and arrays I thought it would be best to omit the betting aspect.

Pseudo Code

System libraries

Functions for shuffle cards , print cards, print hand, next card, score hand, and print all

Main function
    Random number generator
    Declare variables for cards
        Call for shuffle function
    Set functions to deal cards
    Declare variables for player choice
    Declare variable that sets score hand variable

    Do
        Print house cards
        Print dealer cards and total

        Allow player to request or deny new card
            If player says yes, provide new card
        Else if player says no, end game and display scores
        Else display message indicating incorrect entry

        Call for player score function

    While house scores less than 17
        Call for new card for house
        Score house cards

    If house loses

Display message
Print all final results

Else when house doesn't automatically loose

If scores of house and player are equal
Declare tie and print final scores

Else if player score higher than house and less than 21
Display winning message and final score

Else house wins
Display loss message and final scores
End main function

**Definition of void shuffle function*
For card count
Set deck to count to false

**definition of void print card function*
Declare variable value of cards and set to 13 choices

If value of Ace
Else if value of cards 1-9
Else if value of Jack
Else if value of Queen
Else value of king

Declare variable for card suits
If suit if Clubs
If suit is Diamonds
If suit is Hearts
If suits is Spades

**Definition of void print hand function*
For set counter
Declare next card
Print nest card

**Definition of next card function*

Do set new card to random

If deck fails, deal new card

Deal set to false

While card is dealt

Return new card

**Definition of Score hand function*

Set count variables

For card counter

Deal next card and set rank

If return ace

Else if return score plus new card

Else return score plus card with value of 10

While set Ace to 11

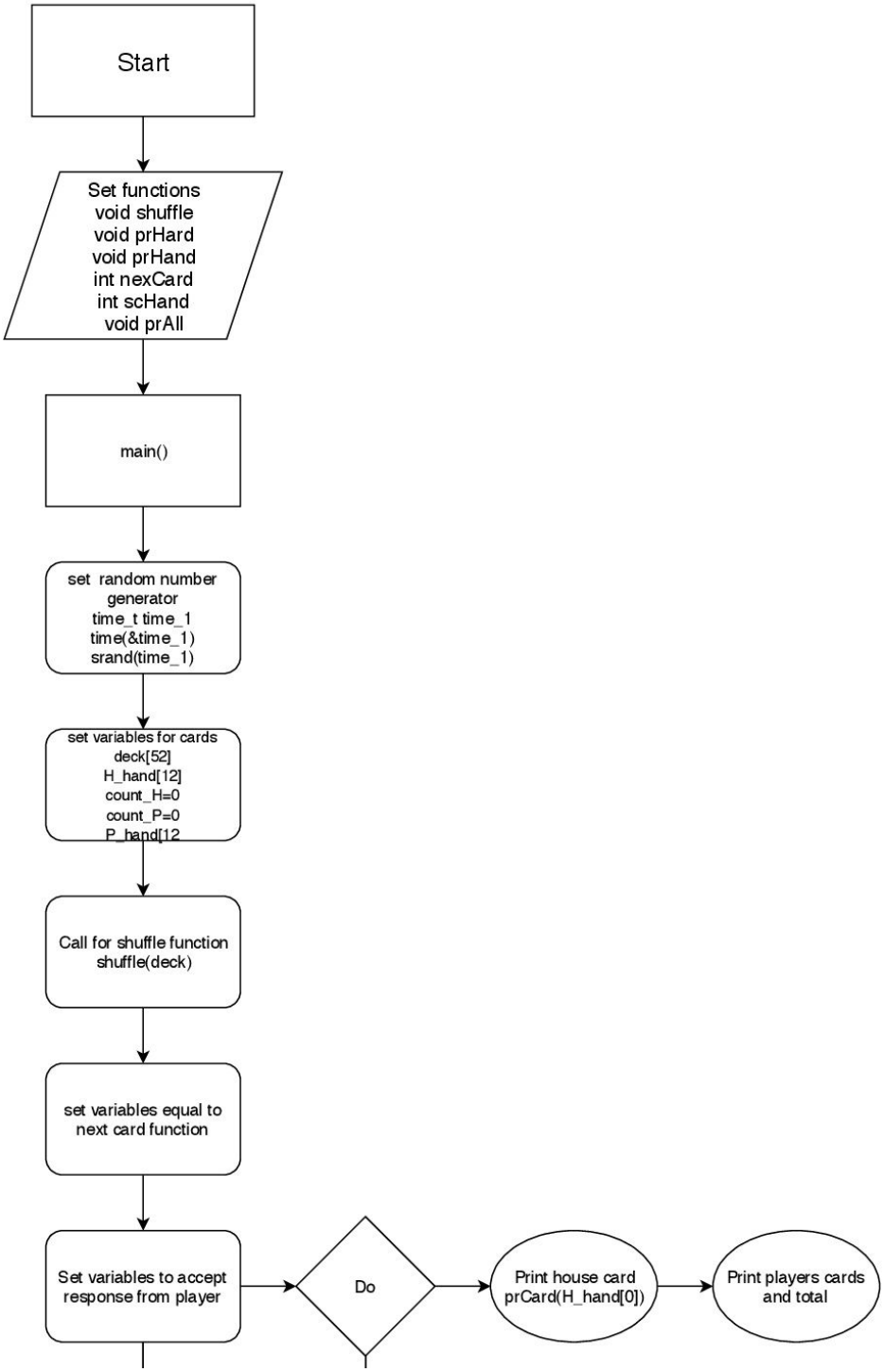
Return score

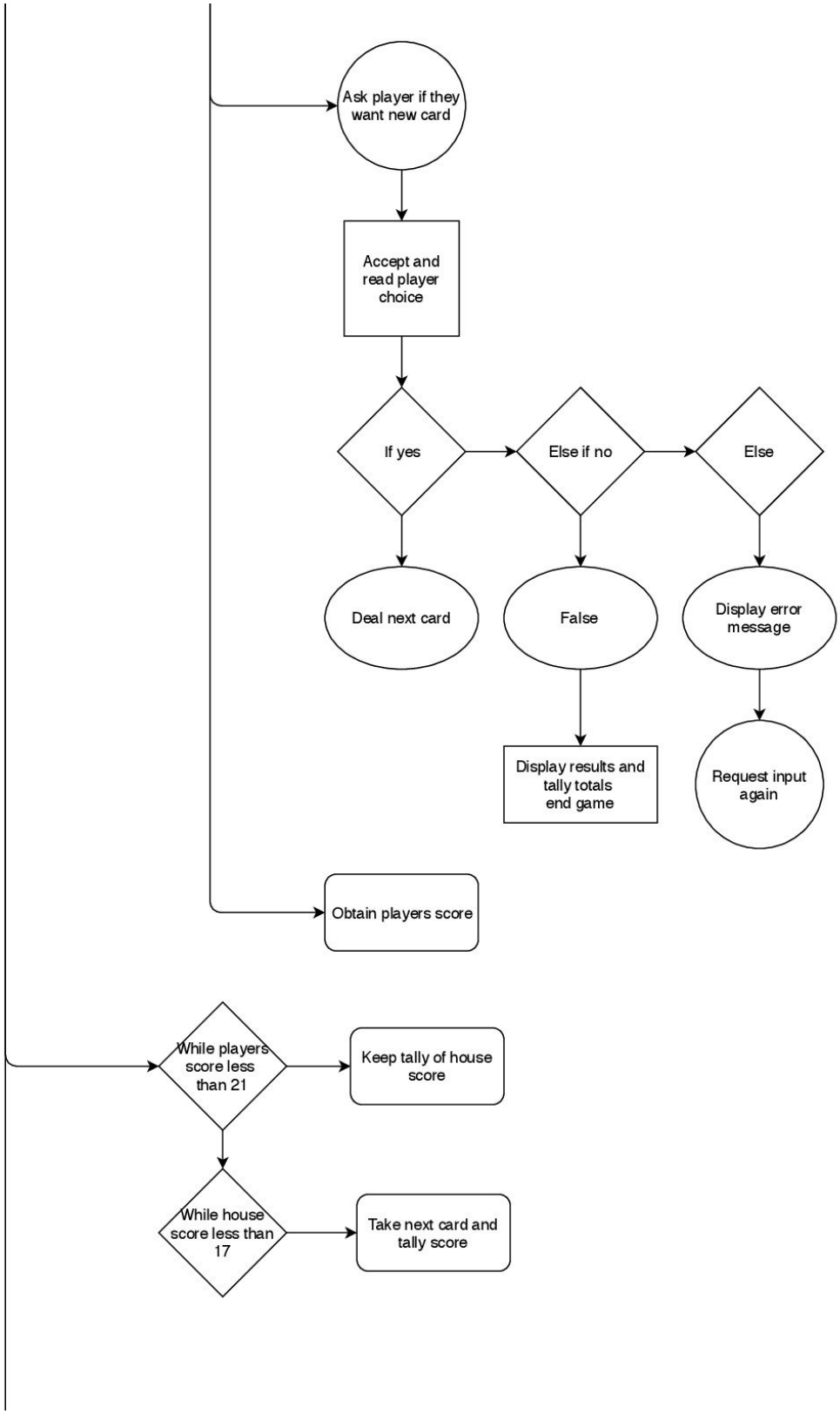
**Definition of void print all function*

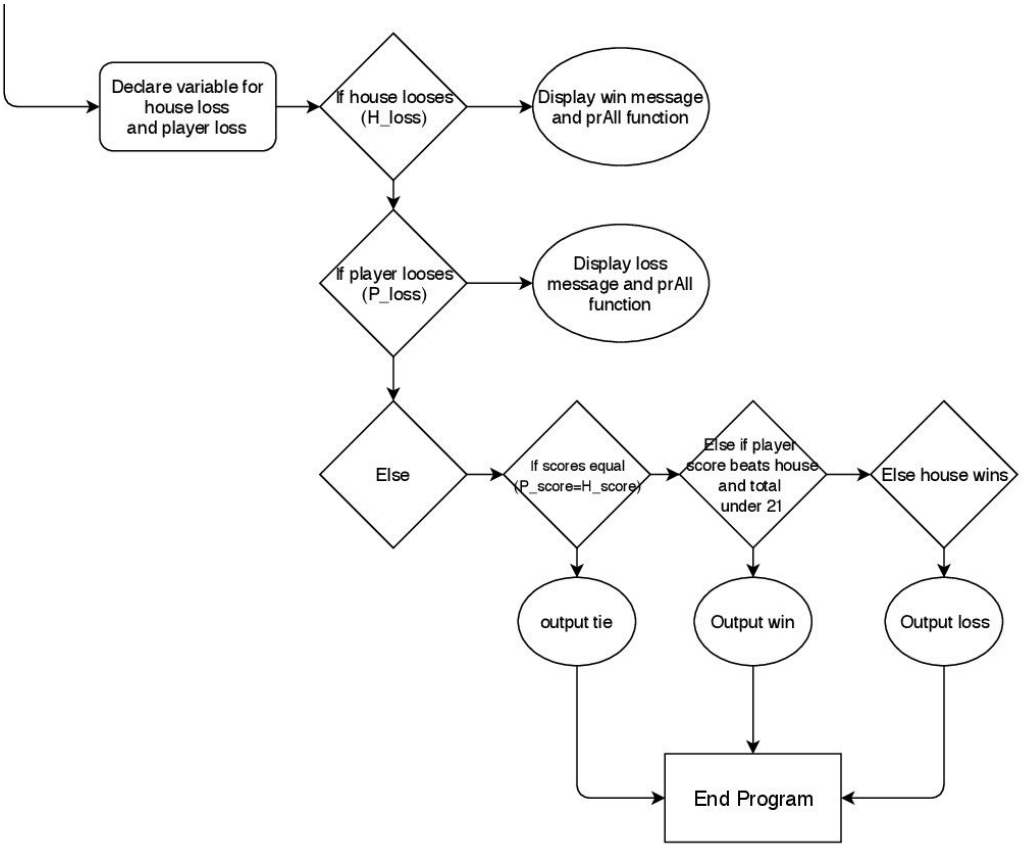
Display house hand and total

Display players hand and total

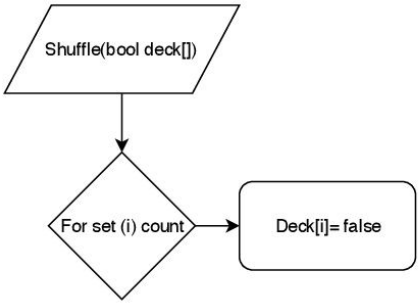
Flowchart


Start
Set functions
void shuffle
void prHard
void prHand
int nexCard
int scHand
void prAll
main()
set random number generator
time_t time_1
time(&time_1)
srand(time_1)
set variables for cards
deck[52]
H_hand[12]
count_H=0
count_P=0
P_hand[12
Call for shuffle function
shuffle(deck)
set variables equal to next card function
Set variables to accept response from player
Do
Print house card
prCard(H_hand[0])
Print players cards and total

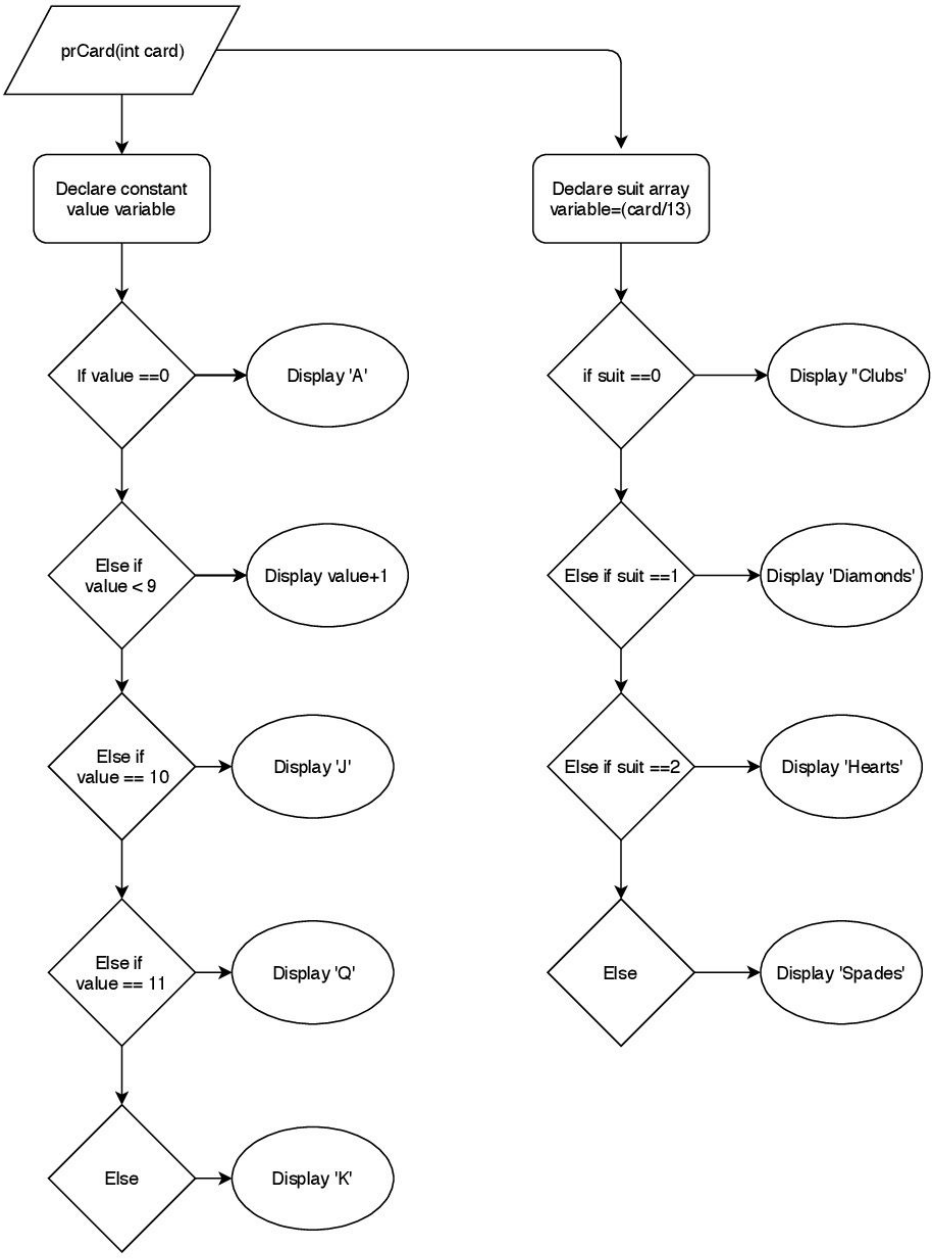

Ask player if they want new card
Accept and read player choice
If yes
Else if no
Else
Deal next card
False
Display error message
Display results and tally totals end game
Request input again
Obtain players score
While players score less than 21
Keep tally of house score
While house score less than 17
Take next card and tally score



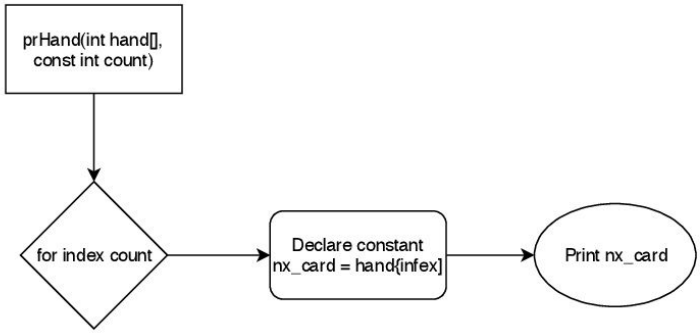
Void shuffle function



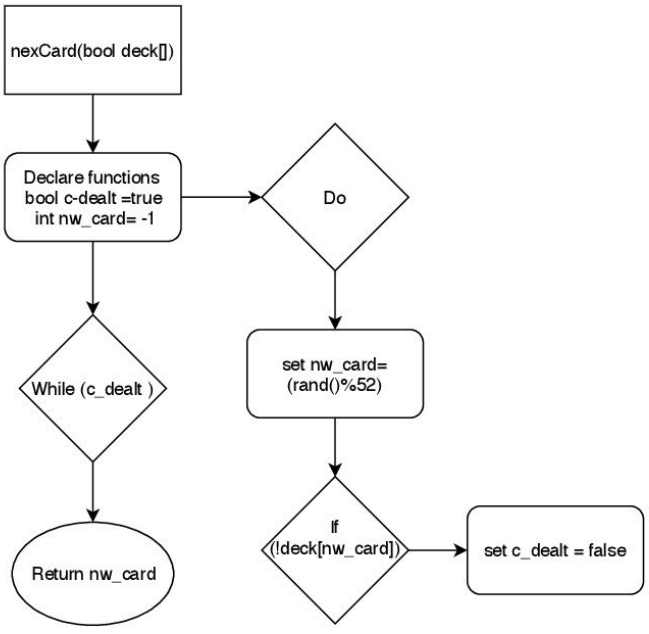
Void prCard function



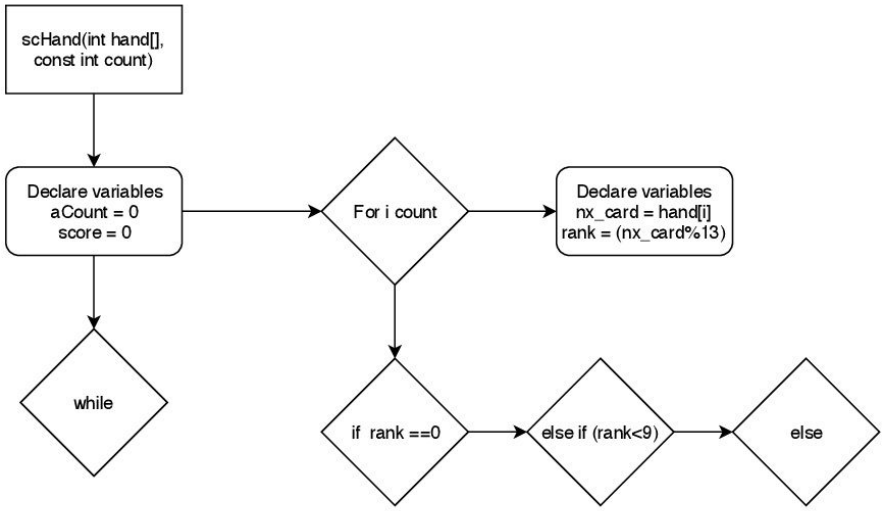
Void prHand Function



nexCard function



scHand Function



Constructs and concepts list

Name	Description	Location	Frequency
libraries	Iostream, ctime	9, 10	2
Integers	Declaring numbers	Throughout	14
Comments	Explaining program	Throughout	57
If	Conditional	77, 117, 124, 159, 181, 220, 240	7
Else-if	Conditional	82, 124, 129, 134, 163, 167, 171,185, 189, 245	10
Else-if-else	Conditional	122-139	1
Logical operators	Make conditions true or false	95, 129, 134, 220, 254	5
Validating user input	Yes or no request for new card	77-85	2
While	Repeat when condition true	95, 100, 254,	3
Do-while	When condition exists, execute	56, 216	2
For loop	Repeat execution when true	148, 202, 235	3
Functions	Main	22	1
Function prototypes	Void and Int	11, 12, 13, 14, 15, 18	6
Pass by value	Pass argument	53, 69, 70, 79, 93, 102, 104	7
Return	Stop and call function	142, 226, 260	3
Returning boolean	Return values	52, 84, 149, 213, 222,	5

Pass by reference	Storage for original argument	43-46, 98, 108, 109, 204	5
Exit function	Exit game	142	1
Single dimension arrays	Holds values	30, 31	2
Single dimension as function arguments	Passing argument	79, 102, 149, 204, 220	5
Passing array to and from functions	Summing arrays	148, 204, 235	3
Searching and sorting arrays	Shuffle and search	149, 220	2

Code

```
#include <iostream>
#include <ctime>

void shuffle(bool deck[]);                  //Shuffle deck function
void prCard(int card);                      //Print card function
void prHand(int hand[], const int count);   //Print hand function
int nexCard(bool deck[]);                   //Deal next card function
int scHand(int hand[], const int count);    //Score of hand function

//Print all hands and scores function
void prAll(int H_hand[], const int H_count, int P_hand[], const int P_count);

using namespace std;

int main()
{
   //Random number generator for card deck
   time_t time_1;
   time(&time_1);
```

```
srand(time_1);

//Variables for cards
bool deck[52];          //Holds deck of 52 cards
int H_hand[12];         //Holds house hands
int countH = 0;         //Holds count of house hands
int countP = 0;         //Holds count of player hands
int P_hand[12];         //Holds players hands


//Initialize game

//Call for shuffle cards
shuffle(deck);

//Set two cards to each house and player
P_hand[0] = nexCard(deck);
H_hand[0] = nexCard(deck);
P_hand[1] = nexCard(deck);
H_hand[1] = nexCard(deck);
countH = 2;
countP = 2;

//Set variables for accepting player choice and dealing player cards
char choice;
bool hits = true;
int P_score = scHand(P_hand, countP);

//accept player choice and deal player cards
do
{
  //Print house cards
   cout << "House Cards\n";
   cout << "-----------------------\n";
   cout << "'Hidden card' & ";
   prCard(H_hand[0]);
   cout << endl;
   cout << endl;
```

```
    //Print dealer cards and score
    cout << "Players Cards\n";
    cout << "------------------------\n";
    prHand(P_hand, countP);
    cout << "Players score: " << scHand(P_hand, countP) << endl;

    //Player new card request
    cout << "Would you like a new card? (y) for yes, (n) for no.\n";
    cin >> choice;
    cout << endl;

    if(choice == 'y')
    {
       P_hand[countP] = nexCard(deck);
       ++countP;
    }
    else if (choice == 'n')
    {
       hits = false;
    }
    else
    {
       cout << "You did not enter 'y' or 'n' please try again.\n";
       cin >> choice;
    }

    //Obtain players score
    P_score = scHand(P_hand, countP);
}
    while (hits && P_score <= 21);

    //Keep tally of score
    int H_score = scHand(H_hand, countH);

    while (H_score < 17)   //Get new card if house card total is less than 17
    {
       H_hand[countH] = nexCard(deck);
       ++countH;
       H_score = scHand(H_hand, countH);
```

```
    }

    //Declare variable for house loss and player loss
    bool H_lost = (H_score > 21);
    bool P_lost = (P_score > 21);

    if(H_lost)  //If house looses
    {
       cout << "**House has lost.**\n";
       cout << "**Congratulations, you win!**\n";
       prAll(H_hand, countH, P_hand, countP);
    }
    if (P_lost)
    {
       cout << "**You went over, sorry, you lose.**\n";
       prAll(H_hand, countH, P_hand, countP);
    }
    else    //If it is a tie
    {
       if (P_score == H_score)
       {
          cout << "**It is a tie!**\n";
          prAll(H_hand, countH, P_hand, countP);
       }
       else if (P_score > H_score && P_score <= 21)  //Player wins
       {
          cout << "**Congratulations, you win!**\n";
          prAll(H_hand, countH, P_hand, countP);
       }
       else if (H_score > P_score && H_score <= 21) //If the house wins
       {
          cout << "**House wins.**\n";
          cout << "**Sorry, you lose!**\n";
          prAll(H_hand, countH, P_hand, countP);
       }
    }

   return 0;
}
```

```
/*******************************************************************************/
/*****************************Shuffle cards*************************************/
void shuffle(bool deck[])
{
   for (int i = 0; i < 52; ++i)
      deck[i] = false;
}
/*******************************************************************************/
/***********************Print cards*********************************************/
void prCard(int card)
{
  //Declare variable
   const int value = (card % 13);

   //Print card values
   if (value == 0)        //Call for Aces
   {
      cout << 'A';
   }
   else if (value < 9)    //Call for all numeric cards
   {
      cout << (value + 1);
   }
   else if (value == 10)   //Call for Jacks
   {
      cout << 'J';
   }
   else if (value == 11)   //Call for Queens
   {
      cout << 'Q';
   }
   else                    //Call for Kings
      cout << 'K';

   //Print car suit
   const int suit = (card/13);

   if (suit == 0)         //Call for Clubs
   {
```

```
      cout << " of Clubs";
   }
   else if (suit == 1)     //Call for Diamonds
   {
      cout << " of Diamonds";
   }
   else if (suit == 2)     //Call for Hearts
   {
      cout << " of Hearts";
   }
   else                    //Call for Spades
   {
      cout << " of Spades";
   }
}
/*****************************************************************************/
/***************************Print hand****************************************/
void prHand(int hand[], const int count)
{
   for (int index = 0; index < count; index++)
   {
      const int nx_card = hand[index];
      prCard(nx_card);
      cout << " " << endl;
   }
}
/*****************************************************************************/
/*************************Deal next card**************************************/
int nexCard(bool deck[])
{
   bool c_dealt = true;       //Set card dealt to true
   int nw_card = -1;          //Initialize new card

   do  //Deal new card when requested
   {
      nw_card = (rand()%52);

      if(!deck[nw_card])
      {
```

```
            c_dealt = false;
        }
    }
    while (c_dealt);
    return nw_card;
}
/******************************************************************************/
/**********************Tally score of individual hands*****************************/
int scHand(int hand[], const int count)
{
    int aCount = 0;         //Ace count
    int score = 0;          //Score count

    for (int i = 0; i < count; ++i)     //Set card counter
    {
        const int nx_card = hand[i];
        const int rank = (nx_card % 13);

        if (rank == 0)          //Return score and Ace card
        {
            ++aCount;
            ++score;
        }
        else if (rank < 9)      //Return score plus value of new card
        {
            score = score + (rank +1);
        }
        else                    //Return score plus card with card value of 10
        {
            score = score + 10;
        }
    }
        while (aCount > 0 && score < 12)    //Set Ace card to 11
        {
            --aCount;
            score = score + 10;
        }
        return score;           //Display score of total hands
}
```

```
/*******************************************************************************/
/************************Total score*****************************************/
void prAll(int H_hand[], const int H_count, int P_hand[], const int P_count)
{
   //Display house final score and hands dealt
   cout << "House hand: " << endl;
         prHand(H_hand, H_count);
   cout << "House score: " <<  scHand(H_hand, H_count) << endl;
   cout << endl;

   //Display players final score and hands dealt
   cout << "Players hand: " << endl;
         prHand(P_hand, P_count);
   cout << "Players score: " << scHand(P_hand, P_count) << endl;
   cout << endl;
}
```