**Project 2**

Title

**Drewbage Cribbage Game V.2**

Course

**CSC-5**

Section

**40651**

Due Date

**Feb 12, 2014**

Author

**Drew Allensworth**

Table of Contents

[Introduction 3](#_Toc379658870)

[Game Play and Rules 3](#_Toc379658871)

[Development Summary 3](#_Toc379658872)

[Flowcharts 3](#_Toc379658873)

[Pseudocode 3](#_Toc379658874)

[Version 1 3](#_Toc379658875)

[Version 2 8](#_Toc379658876)

[Function Descriptions and Comments 8](#_Toc379658877)

[Version 1 8](#_Toc379658878)

[Version 2 14](#_Toc379658879)

[Major Variables and Objects 14](#_Toc379658880)

[Version 1 14](#_Toc379658881)

[Version 2 14](#_Toc379658882)

[Concepts Used 14](#_Toc379658883)

[Program Listing 14](#_Toc379658884)

**Drewbage** *Cribbage... Distilled.*

# Introduction

At the outset of this project, I knew that one of the most difficult games for me to implement with any modicum of success would be a card game. The difficulty, as I saw it, would be in attempting to accurately model the behavior of having cards in one’s hand and then using them. That is, to assign cards to a hand from a “deck”—insuring no duplicates could go to any player—and then removing those cards from one’s hand to use in gameplay. Even if that was implemented successfully, I still needed to create ways to count the points in a hand, allowing the player to use the hand to their advantage. Despite the difficulty, I desired to impress. In addition, Cribbage holds a special place for me as it’s the only card game that my family played during my childhood and still continues to play. It’s the game that I would go to my grandmother’s house to play before she became ill. Any day that we would travel to her house, we would wait till after dinner and her, my family, and I would play cribbage together.

# Game Play and Rules

Drewbage cribbage is two-player cribbage.

# Development Summary

# Flowcharts

# Pseudocode

## Version 1

***Four parallel arrays\**** *Example:*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sub pos. | Sub 0 | Sub 1 | Sub 2 | Sub 3 | Sub 4 | Sub 5 |
| crdID | 0 | 1 | 2 | 3 | 4 | 5 |
| crdName | “a\_c” | “a\_d” | “a\_h” | “a\_s” | “2\_c” | “2\_d” |
| crdSuit | 1 | 2 | 3 | 4 | 1 | 2 |
| crdVal | 1 | 1 | 1 | 1 | 2 | 2 |

\*crdName must be a vector to hold strings

**Pseudocode**

1. ~~Display splash graphics~~
2. Display menu
   1. 1. Play game
   2. ~~2. Help~~
   3. ~~3. High Scores~~
   4. Any other key to Quit
3. Get menu choice
4. (Assuming case == “Play Game”)
   1. Initialize and fill the card arrays and vector for card ID, Values, Suits, and Names
   2. Initialize playerPoints=0, computerPoints=0
   3. Have the computer randomly determine who gets dealt first and who gets “nibs”. Store the outcome dealer.
      1. Alert the player
   4. Deal the player and the computer six cards each
      1. Start with a “deck” of 52 unique cards
         1. If dealer is for the player
            1. Loop for six times

Randomly pick a card ID for the player

Insure that card hasn’t been chosen

Assign that card ID to the player’s “hand”

Remove the possibility of that card being picked again

Pick a card ID for the computer

Insure that card hasn’t been chosen

Assign that card ID to the computer’s “hand”

Remove the possibility of that card being picked again

* + - * 1. End loop
      1. Else dealer is for the computer
         1. Loop for six times

Pick a card ID for the computer

Insure that card hasn’t been chosen

Assign that card ID to the computer’s “hand”

Remove the possibility of that card being picked again

Randomly pick a card ID for the player

Insure that card hasn’t been chosen

Assign that card ID to the player’s “hand”

Remove the possibility of that card being picked again

* + - * 1. End loop
  1. For each card in the player’s “hand”, display ~~a graphic of the card and~~ the card’s name, i.e. ace\_S or queen\_H (ace of spades and queen of hearts respectively)
     1. ~~Using the card name and files that separately store each card’s graphic~~
     2. Display the card name which the player will use to reference the card ~~next to the image~~
  2. Randomly select a card ID as the starter card from the “deck” and store the selection
     1. Remove the possibility of that card being picked again
     2. Display ~~a graphic of the card and~~ the card’s name
     3. If the card is a Jack
        1. Give two points to either the player or computer based on dealer
        2. Alert the player of who got the points
  3. Have the player discard two cards
     1. Loop two times
        1. Show Player’s “hand”
        2. Ask player to type a card name to discard
        3. Use the card name input to find the number ID
           1. Loop –If card does not exist, alert player and get card name again
           2. Loop –If card is not in player’s “hand”, alert player and get card name again
        4. Assign card ID to crib “hand”
        5. Remove the ID from the player’s “hand”
     2. End loop
  4. Have the computer discard two cards
     1. Loop two times
        1. Randomly select a card to discard using the sub values
           1. Loop –If card ID is not valid, select another sub value
        2. Assign card ID to crib “hand”
        3. Remove the ID from the computer’s “hand”
     2. End loop
  5. Start The Play
     1. Make a copy of the Player’s “hand”
     2. Make a copy of the computer’s “hand”
     3. Initialize unsigned short playCnt=0 as the variable to hold the count for the gameplay
     4. Initialize bool computerPass=false, playerPass=false
     5. Declare char lastPlay//valid values are ‘p’ for player and ‘c’ for computer
        1. If dealer is for the player—start with the computer
           1. Loop do {

Have the computer select a card from “hand” to use using the sub values

Validate: Loop –Start at sub 0. If card ID <0, move to next sub value < vector.size()

Validate: Loop –if playCnt + the card selection > 31, move to next sub value < vector.size()

Validate: If no card meets these criteria , display that the computer has passed and return computerPass=true to allow player to select a card

If computerPass=false

Display the card name of computer’s selection

Remove the selected card from the computer’s “hand”

Set playCnt = playCnt + crdVal

If playCnt == 15 then give the computer 2 points

If playCnt == 31 then give the computer 2 points

Set lastPlay=’c’

Display playCnt

Let the Player select a card

Display Player’s “hand”

Ask player to type a card name to use

Use the card name input to find the number ID

Validate: Loop –If card does not exist, alert player and get card name again

Validate: Loop –If card is not in player’s “hand”, alert player and get card name again

Validate: Loop –if playCnt + the card selection > 31, alert player and get card name again

If player types ‘P’, set playerPass=true

If playerPass=false

Remove the selected card from the Player’s “hand”

Set playCnt = playCnt + crdVal

If playCnt == 15 then give the Player 2 points

If playCnt == 31 then give the Player 2 points

Set lastPlay=’p’

Display playCnt

Display Player’s “hand”

* + - * 1. } while(playCnt<31&&(playerPass==false||computerPass==false)
      1. If dealer is for the computer—start with the player
         1. Loop do {

Display Player’s “hand”

Display playCnt

Ask player to type a card name to use

Use the card name input to find the number ID

Validate: Loop –If card does not exist, alert player and get card name again

Validate: Loop –If card is not in player’s “hand”, alert player and get card name again

Validate: Loop –if playCnt + the card selection > 31, alert player and get card name again

If player types ‘P’, set playerPass=true

If playerPass=false

Remove the selected card from the Player’s “hand”

Set playCnt = total + crdVal

If playCnt == 15 then give the Player 2 points

If playCnt == 31 then give the Player 2 points

Set lastPlay=’p’

Display playCnt

Display Player’s “hand”

Let the computer select a card

Have the computer select a card to use using the sub values

Validate: Loop –Start at sub 0. If card ID <0, move to next sub value < vector.size()

Validate: Loop –if playCnt + the card selection > 31, move to next sub value < vector.size()

Validate: If no card meets these criteria , display that the computer has passed and return computerPass=true to allow player to select a card

If computerPass=false

Display the card name of computer’s selection

Remove the selected card from the computer’s “hand”

Set playCnt = playCnt + crdVal

If playCnt == 15 then give the computer 2 points

If playCnt == 31 then give the computer 2 points

Set lastPlay=’c’

Display playCnt

* + - * 1. } while(playCnt<31&&(playerPass==false||computerPass==false)
      1. If(playerPass==true && computerPass==true)
         1. If lastPlay==’p’ then give the player one point
         2. Else give the computer one point
      2. While ((Player still has cards) ||(computer still has cards)){
         1. This time, instead of choosing who goes first based on dealer, it’s based on the value in lastPlay.
         2. Reset computerPass = False and playerPass = False
         3. Start “The Play” above
         4. }
    1. Output scores for both the computer and the player
       1. If there is a nibs bonus, apply it to the dealer
    2. Calculate winner
       1. If playerPoints>computerPoints then Player wins
       2. Else computer wins
  1. ~~Get a player’s name or handle~~
  2. ~~Store name in a file with the score~~

1. ~~End of case “Play Game”~~
2. ~~(Assuming case == “Help”)~~
   1. ~~Use an outputfile object to output help text~~
3. ~~(Assuming case == “High scores”)~~
   1. ~~Use an outputfile object to output all scores in the scores file~~

## Version 2

**Pseudocode**

1. Create card structure
   1. Int ID
   2. Int Rank
   3. Int Suit
   4. String Name
   5. Int Val
2. Create and initialize the “sets” of possible card combinations for scoring
3. Create an array with 52 card structure elements
4. Fill the crdDeck array element members from files
5. Using “hands” passed from plyGame()
   1. Create copies to use for the program
6. Make copies of the copied hands and sort them in ascending order
7. For each hand, take the sorted hand and loop through the array filling an array of card structure elements based on the ids in the sorted hand.
8. Create copies of both the player’s hand, the computer’s hand, and the crib with the up card added.
9. Based on dealer, start with either Player first or computer
   1. Score hands for pairs
   2. Score hands for 15s
      1. Sets of two
      2. Sets of three
      3. Sets of four
      4. Sets of five
   3. Score hands for runs
      1. Start with runs of five—if found stop.
      2. Then Four—if found stop.
      3. Then Three
   4. Score hands for flush
      1. If even one card is different, no flush
   5. Score hands for nibs
      1. must figure this one out
10. Score the crib with the same methods as above.
    1. Assign points to dealer
11. Output scores for both the computer and the player
    1. If there is a nibs bonus, apply it to the dealer
12. Calculate winner
    1. If playerPoints>computerPoints then Player wins
    2. Else computer wins
13. Get a player’s name or handle
14. Store name in a file with the score
15. End of case “Play Game”
16. (Assuming case == “Help”)
    1. Use an outputfile object to output help text
17. (Assuming case == “High scores”)
    1. Use an outputfile object to output all scores in the scores file

# Function Descriptions and Comments

## Version 1

|  |  |
| --- | --- |
| void filName(vector<string> crdName, ifstream &names) | Parameters: vector<string> crdName is an empty vector from plyGame(); ifstream &names is a files with string names  Uses names file to fill the crdName vector |
| void filIds(short []crdID, ifstream &ids,const short DK\_SIZE) | Parameters: empty short array sized to DK\_SIZE, ifstream object connected to a file of short ints; const short DK\_SIZE is the size of the array  Uses the referenced file to fill the crdID array with values |
| void filVals(short []crdVal, ifstream &vals,const short DK\_SIZE) | Parameters: empty short array sized to DK\_SIZE, ifstream object connected to a file of short ints; const short DK\_SIZE is the size of the array  Uses the referenced file to fill the crdVal array with values |
| void filSuit(short []crdSuit, ifstream &suits,const short DK\_SIZE) | Parameters: empty short array sized to DK\_SIZE, ifstream object connected to a file of short ints; const short DK\_SIZE is the size of the array  Uses the referenced file to fill the crdSuit array with values |
| void plyGame() | Main function for gameplay |
| char GtDealr() | Uses rand() to generate a character which will decide who gets dealt first—the computer or the Player. ‘C’ for Comp; ‘P’ for Player. The one who gets dealt first also gets “nibs”. This information will be outputted as a message.  Output: A char with ‘P’ for player and ‘C’ for computer |
| void deal(short deck[], vector<short> &handC, vector<short> &handP ,char dealer) | Parameters: deck[],vector<short> &handC, vector<short> &handP, and dealer from plyGame().  Uses and changes the deck array, the computer’s hand vector, and the player’s hand vector using addCard(). Char dealer contains ‘P’ for player or ‘C’ for computer. Either starts dealing to the computer or Player based on the value of dealer. Copies the value to a hand, and then changes it in the deck to -1 to indicate that it was used. Loops for six cards to each player. |
| short addCard(short deck[], vector<short> &) | Parameters: deck[] from plyGame(); vector<short> is an empty vector  Uses and the deck array. Changes the computer’s hand vector, and the player’s hand vector. Uses SlCrdDk() for array sub number to select. Uses inDeck() to determine if card is available.  Output: card ID between 0-51 |
| short SlCrdDk() | Uses rand() to select a number between 0 and 51, that value is returned as a short. The number is used for an array sub number |
| bool inDeck(short cardSub, const short[]) | Parameters: cardSub is a short between 0-51 and the short[] is the deck array from plyGame().  Uses cardSub to determine if the card is available for selection or has already been selected  Output: boolean, true if card is available; otherwise false |
| bool inHand(short id, const vector<short> &) | Parameters: cardSub is a short between 0-51 and the vector is a “hand” with card ID’s  Uses id to search through the vector to determine if the card is available for selection  Output: boolean, true if card is available; otherwise false |
| bool inHndC(short sub, const vector<short> &hand) | Parameters: sub is a short between 0-51 and the vector is the computer’s hand from plyGame().  Uses sub to determine if the card is available for selection or has already been selected. Used for computer’s hand only.  Output: boolean, true if card is available; otherwise false |
| short cardID(string, const vector<string> &crdName, const short crdID[], const short DK\_SIZE) | Parameters: string is a card name; const string crdName[] from plyGame(); const short crdID[] from plyGame(); const short DK\_SIZE for looping through array  First validates that the string name is a string in the array. If yes, it returns the card ID value in the corresponding sub position of crdID[]. If no, returns -1 to indicate that the card name does not exist  Output: short card ID between 0-51 or if invalid card name -1 |
| void dspHand(vector<short> handP , vector<string> crdName) | Parameters: vector handP is the player’s hand containing card ID’s; vector<string> crdName from plyGame() is a vector containing string names for cards.  Uses the values from the handP vector to look up values in the crdName[] array and displays each card ID which is > 0 |
| short strtCrd(short deck[],bool &isStJak) | Parameters: uses deck[] from plyGame() which holds available card values; uses reference to boolean isStJak from plyGame().  Uses SlCrdDk() to get a random value. Validates that the corresponding card is available using inDeck(). Assigns a value of -1 in the deck to indicate that it was used. Checks if the card equals values for Jacks, if so isStJak=True, else false.  Output: Returns a short containing a card ID |
| void dspHand(short, vector<string> crdName) | Parameters: the short is a card ID from 0-51; vector<string> crdName from plyGame() is a vector containing string names for cards.  !OVERLOADED FUNCTION! Uses the short to display the name for the card ID. |
| void rmveCrd(short, const vector<short> &) | Parameters: the short is a card ID 0-51; vector<short> is any vector being used as a hand with card values  Searches through the vector for the card ID and changes it to -1, effectively removing the card from the hand. |
| void discrdP(vector<short> &handP,vector<short> &crib, const vector<string> crdName , const short cardID[]) | Parameters: vector<short> &handP is Player’s hand from plyGame(); vector<short> &crib from plyGame();const vector<string> crdName from plyGame() used for dspHand() and cardID(); const short cardID[] from plyGame() used for cardID()  Loops twice. Displays Player’s hand using dspHand(). Gets a card name from Player, validates and gets card ID from cardID(). Validates that card is in Player’s hand using inHand(). Assigns card ID to vector<short> &crib. Changes card ID to -1 in Player’s hand using rmveCrd(). |
| void discrdC(vector<short> &handC,vector<short> &crib, const cardID[]) | Parameters: vector<short> &handC is computer’s hand from plyGame(); vector<short> &crib is crib hand from plyGame(); const cardID[] is card ID’s from plyGame()  Loops twice. Randomly selects a card to discard to crib hand. Uses rand()%6 to select a value 0-5 to use as a sub number. Validates that sub number has a valid card ID using inHand(). Assigns card ID to vector<short> &crib. Changes card ID to -1 in computer’s hand. |
| void copyVec(vector<short> &, vector<short> &) | Parameters: two vectors of the same datatype  Takes the values in the first vector and copies them to the second vector. |
| short cardVal(short, const short crdVal[]) | Parameters: a short int card ID from 0-51; crdVal[] is the array of card values from plyGame()  Uses the card ID to determine the card’s value  Output: a short with the value 1-10 |
| void plyCrdC(vector<short> &c\_handC, const short crdVal[], const vector<string> &crdName, unsigned short &playCnt, bool &cmpPass, int &pointsC, char &lastPly) | Parameters: vector<short> &c\_handC is a copy of the computer’s hand from plyGame(); const crdVal[] is an array of card values from plyGame() for cardVal(); const vector<string> crdName[] is an vector of card names for dspHand() from plyGame(); unsigned short &playCnt has value 0-31 from plyGame(); bool &cmpPass initialized to false from plyGame(); int &pointsC is points total from plyGame(); char &lastPly holds ‘C’ or ‘P’ from plyGame()  Used to have the computer play a card from its “hand”. Loops through the hand looking for a card ID >=0. Uses cardVal() to get the card’s value. Checks if card’s value plus playCnt > 31. Returns cmpPass=True if loop ends without selecting a card. If cmpPass=False—displays card using dspHand(), assigns -1 to card ID in hand, calculates playCnt, and sets lastPly = ‘C’. Displays playCnt. |
| void plyCrdP(vector<short> &c\_handP, const short crdID[], const short crdVal[], const vector<string> &crdName, unsigned short &playCnt, bool &plrPass, int &pointsP, char &lastPly) | Parameters: vector<short> &c\_handP is a copy of the Player’s hand from plyGame(); const short crdID[] is an array of card id’s for cardID() from plyGame(); const crdVal[] is an array of card values from plyGame() for cardVal();const vector<string> crdName is a vector of card names for dspHand() from plyGame(); unsigned short &playCnt has value 0-31 from plyGame(); bool &plrPass initialized to false from plyGame(); int &pointsP is points total from plyGame(); char &lastPly holds ‘C’ or ‘P’ from plyGame()  Used to have Player play a card from their hand. Displays Player’s hand using dspHand(). Gets a card name from Player, validates and gets card ID from cardID(). Validates that card is in Player’s hand using inHand().Uses cardVal() to get the card’s value. Checks if card’s value plus playCnt > 31. Sets plrPass=True if Player enters “P”. If plrPass=False—assigns -1 to card ID in hand, calculates playCnt, and sets lastPly = ‘C’. Displays playCnt. Displays card using dspHand(). |
| short vldPlay(const vector<short> &hand, const short crdVal[],const short playCnt) | Parameters: vector<short> hand is a hand containing values -1 through 51; const crdVal[] is an array of card values from plyGame(); unsigned short &playCnt has value 0-31 from plyGame()  Determines if there is a playable card in a particular hand given the current playCnt.  Output: The card ID of the first playable card in the hand, else -1 if no playable cards are found |
| bool hasCrds(vector<short>) | Parameters: Uses a short int vector  Checks each value in the vector to see if any one value is >=0 meaning there is a valid card left to play  Output: returns true if there are one or more cards left, else returns false |
| void thePlay((vector<short> &c\_handC,vector<short> &c\_handP,const short crdVal[], const short crdID[],  const vector<string> &crdName, unsigned short &playCnt,  bool &cmpPass, bool &plrPass, int &pointsC, int &pointsP, char &dealer, char &lastPly, char option) | Parameters: vector<short> &c\_handP is a copy of the Player’s hand from plyGame();vector<short> &c\_handC is a copy of the computer’s hand from plyGame(); const short crdID[] is an array of card id’s for cardID() from plyGame(); const crdVal[] is an array of card values from plyGame() for cardVal();const vector<string> crdName is a vector of card names for dspHand() from plyGame(); unsigned short &playCnt has value 0-31 from plyGame(); bool &plrPass initialized to false from plyGame();int &pointsC is computer’s points total from plyGame(); int &pointsP is player’s points total from plyGame(); char &lastPly holds ‘C’ or ‘P’ from plyGame(); option controls which variable controls the execution order  thePlay() executes plyCrdC() and plyCrdP() using looping constructs. The order of execution is based on either char dealer or char lastPly using char option to determine which variable will be used. Checks if either the player or computer has “passed” and keeps track of the playCnt. Adds “GO” scoring functionality based on lastPly. |

## Version 2

|  |  |
| --- | --- |
| void banner(char []) | Parameters: character array for the filename  Displays banner character image from a file |

# Major Variables and Objects

## Version 1

## Version 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Major Variables and Objects** | | | |
| **Function Location** | **Type** | **Name** | **Description** |
| plyGame() | char array[25] | fName | An array of size 25 to hold the characters of a filename. |

# Concepts Used

|  |  |
| --- | --- |
| **Concepts Used-References Gaddis Chapters** | |
| Chapter 2 | cout object  #include  variables  integer data type  character data type  c++ string class  floating point data type  bool data type  arithmetic operators  comments  named constants |
| Chapter 3 | cin object  type conversion  type casting  mathematical expressions |
| Chapter 4 | Relational operators  if statement  if/else  nested if statements  if/else if  logical operators  menus  validating user input  comparing characters and strings  the conditional operator  switch statement |
| Chapter 5 | while loop  do-while loop  for loop  nested loops  using files for data  breaking a loop |
| Chapter 6 | defining and calling functions  function prototypes  sending data into a function  passing data by value  return statement  reference variables  overloading functions |
| Chapter 7 | array initialization  parallel arrays  processing array contents  arrays as function arguments  STL vector |
| Chapter 8 | searching arrays  searching vectors |
| Chapter 10 | C-strings  library functions for C-strings (strcpy()) |

# Program Listing