Blackjack

Justin Solis

CSC 5

Winter 2017

40652

Table of Contents

Introduction 3

Rules 3

Terminology

Win Conditions

Loss Conditions

Basic Run Through of a Game

Summary of Development 4

Pseudocode 6

Checklist 7-11

Flowcharts 12-17

**Introduction:**

I picked Blackjack as my game because it was the game that I picked for Project 1. Blackjack was one of the games on the list that I felt that I understood the rules well enough to be able to create a game out of it. Blackjack is a fairly simplistic game. The goal of the game is to get above the dealer, and as close to the number 21 as possible.

**Rules:**

Terminology

* Hit = get another card from the deck.
* Bust = have a total higher than 21.
* Stay = player chooses not to receive any more cards, staying at whatever their number is when they choose to hit.

Win conditions

* Player has more than the dealer, and is below or at 21.
* Player is below or at 21, and the dealer busts.

Loss conditions

* Player has a lower total than the dealer.
* Player busts, and the dealer has a lower total than the player.
* Player hand is equal to dealer hand (even when both players bust).

Basic run through of a game

* Player and dealer both get two cards, one of the dealer’s cards is hidden.
* Dealer asks the player to hit or stay.
* If the player hits, then the player receives an extra card.
* If the player stays, they keep the total they have and the dealer’s cards are shown.
* Player either wins or loses based on if they achieve the win/loss conditions.

**Summary of Development:**

Project size: 278 lines

# of variables: ~13 (not including variables initialized for loops)

Starting

I started this project by rereading my write-up for Project 1, and looking at the code for that project. I saw that I said it would be easier if I used more for statements, so I added those to count how many cards the player hits. I basically redid the entire program, because I did not like the baseline the original program set. I kept some things from the original program, like using a while statement to control most of the game. I also separated most of the program into different functions. For example, I created functions for: the main game, shuffling cards, outputting win/loss, and to end the game.

Problems

I ran into some problems at the end of development, because some cards were outputting incorrectly when the player wins/loses the game. An example would be: in the main function card 4 would equal 10, but in the win/loss functions card 4 would equal 5. This problem only happened after the 4th card in the array was output. Since I was unable to figure out what was causing the problem, I used two if statements to bandage the problem. One if statement checked if the amount of cards being output was less than or equal to 4. If it was, then it would output those 4 cards and the total of the hand normally. The other if statement checked if the amount of cards being output was greater than or equal to 4. If it was, then it would only output the total of the hand.

Inputs

Similar to the last project, the inputs are self-explanatory. Pressing 1 starts the program and gives the player more cards, and pressing 2 ends the game.

**Pseudocode**

Int main

Initialize variables

State what the game will do

Ask the user to press 1 or 2 to either start the game or end the program

If the user presses 1, start the game

Call game function

If the user presses 2, end the game

Call end function

End game

Int game

Initialize variables

Call the shuffle function

Output the player’s and dealer’s cards

Use a while loop to control the game

Ask the user if they want to hit

If they hit, output another card

If they stay, go to the win function or loss function

Int shuffle

Set random number seed

Set the cards as random, and make card>10=10

Int win

Setup output file

Initialize variables

Output the user’s hand

Output the dealer’s hand

State the player won

Output results to file

Int loss

Setup output file

Initialize variables

Output the user’s hand

Output the dealer’s hand

State the player won

Output results to file

Bool end

Return false to end game

**Cross Reference from Project 1**

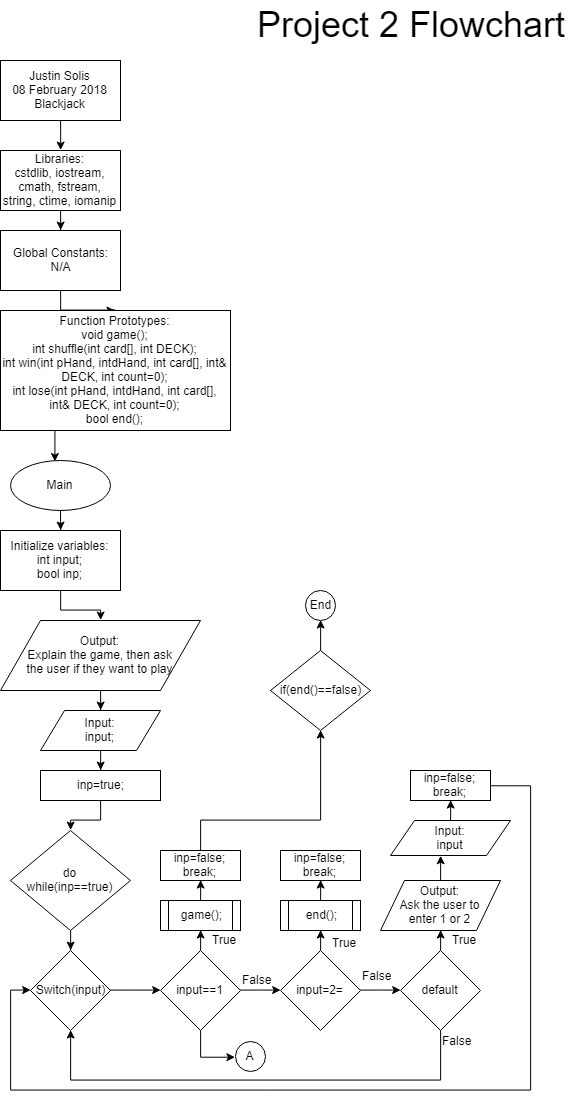
**You are to fill-in with where located in code**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Chapter** | **Section** | **Topic** | **Where Line #''s** | **Pts** | **Notes** |
| 2 | 2 | cout |  |  |  |
|  | 3 | libraries | 9-15 | 5 | iostream, iomanip, cmath, cstdlib, fstream, string, ctime |
|  | 4 | variables/literals |  |  | No variables in global area, failed project! |
|  | 5 | Identifiers |  |  |  |
|  | 6 | Integers | 20-22 | 1 | Used throughout |
|  | 7 | Characters |  | 1 |  |
|  | 8 | Strings | 207, 248 | 1 |  |
|  | 9 | Floats No Doubles | 65 | 1 | Using doubles will fail the project, floats OK! |
|  | 10 | Bools | 23 | 1 |  |
|  | 11 | Sizeof \*\*\*\*\* |  |  |  |
|  | 12 | Variables 7 characters or less |  |  | All variables <= 7 characters |
|  | 13 | Scope \*\*\*\*\* No Global Variables |  |  |  |
|  | 14 | Arithmetic operators |  |  |  |
|  | 15 | Comments 20%+ |  | 2 | Model as pseudo code |
|  | 16 | Named Constants |  |  | All Local, only Conversions/Physics/Math in Global area |
|  | 17 | Programming Style \*\*\*\*\* Emulate |  |  | Emulate style in book/in class repositiory |
|  |  |  |  |  |  |
| 3 | 1 | cin |  |  |  |
|  | 2 | Math Expression |  |  |  |
|  | 3 | Mixing data types \*\*\*\* |  |  |  |
|  | 4 | Overflow/Underflow \*\*\*\* |  |  |  |
|  | 5 | Type Casting | 185 | 1 |  |
|  | 6 | Multiple assignment \*\*\*\*\* |  |  |  |
|  | 7 | Formatting output | 212-213, 225-226, 230, 233 | 1 | Also used in the loss function |
|  | 8 | Strings | 232, 235, 273, 278 | 1 |  |
|  | 9 | Math Library | 11 | 1 | All libraries included have to be used |
|  | 10 | Hand tracing \*\*\*\*\*\* |  |  |  |
|  |  |  |  |  |  |
| 4 | 1 | Relational Operators |  |  |  |
|  | 2 | if | 105 | 1 | Independent if, used throughout |
|  | 4 | If-else |  | 1 |  |
|  | 5 | Nesting | 102-148 | 1 |  |
|  | 6 | If-else-if | 79-90 | 1 | Used throughout |
|  | 7 | Flags \*\*\*\*\* |  |  |  |
|  | 8 | Logical operators | 79 | 1 | Used throughout |
|  | 11 | Validating user input | 40-49 | 1 | The while statement later in the program continually asks the user if they want to hit as well. |
|  | 13 | Conditional Operator |  | 1 |  |
|  | 14 | Switch | 41-48 | 1 |  |
|  |  |  |  |  |  |
| 5 | 1 | Increment/Decrement | 104 | 1 | Used throughout |
|  | 2 | While | 93 | 1 |  |
|  | 5 | Do-while | 40-49 | 1 |  |
|  | 6 | For loop | 104-146 | 1 | Used throughout |
|  | 11 | Files input/output both | 203-204, 244-245 | 2 |  |
|  | 12 | No breaks in loops \*\*\*\*\*\* |  |  | Failed Project if included |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| \*\*\*\*\*\* Not | required to | show | Total | 30 |  |

**Cross Reference for Project 2**

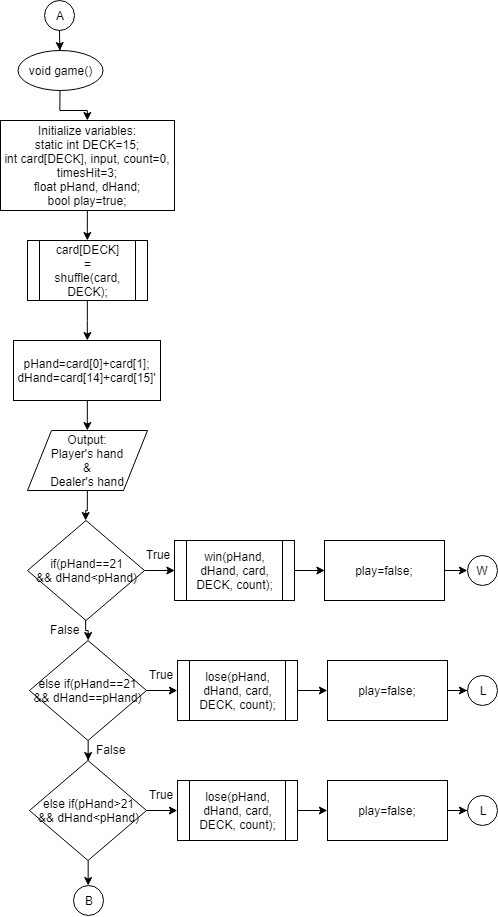
**You are to fill-in with where located in code**

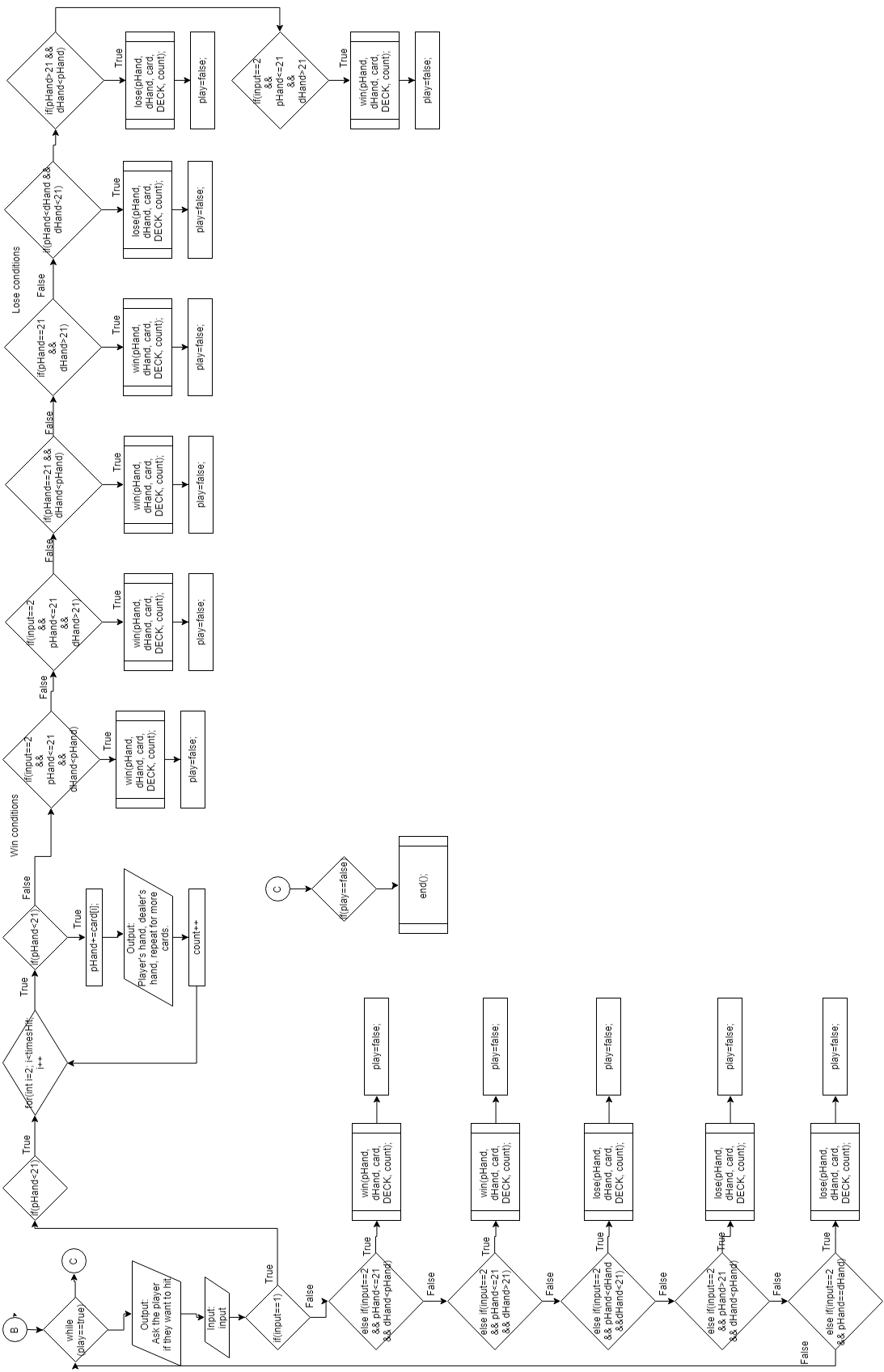
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Chapter** | **Section** | **Topic** | **Where Line #''s** | **Pts** | **Notes** |
| 6 |  | Functions |  |  |  |
|  | 3 | Function Prototypes | 19-23 | 4 | Always use prototypes |
|  | 5 | Pass by Value |  | 4 |  |
|  | 8 | return | 195-286 | 4 | A value from a function |
|  | 9 | returning boolean | 286 | 4 |  |
|  | 10 | Global Variables |  | XXX | Do not use global variables -100 pts |
|  | 11 | static variables | 62 | 4 |  |
|  | 12 | defaulted arguments | 21, 22 | 4 |  |
|  | 13 | pass by reference | 21, 22, 201, 242 | 4 |  |
|  | 14 | overloading |  | 5 |  |
|  | 15 | exit() function | 52 | 4 |  |
| 7 |  | Arrays |  |  |  |
|  | 1 to 6 | Single Dimensioned Arrays | 63 | 3 |  |
|  | 7 | Parallel Arrays |  | 2 |  |
|  | 8 | Single Dimensioned as Function Arguments 69 | | 2 |  |
|  | 9 | 2 Dimensioned Arrays |  | 2 | Emulate style in book/in class repositiory |
|  | 12 | STL Vectors |  | 2 |  |
|  |  | Passing Arrays to and from Functions 69, 195 | | 5 |  |
|  |  | Passing Vectors to and from Functions | | 5 |  |
|  |  |  |  |  |  |
| 8 |  | Searching and Sorting Arrays |  |  |  |
|  | 3 | Bubble Sort |  | 4 |  |
|  | 3 | Selection Sort |  | 4 |  |
|  | 1 | Linear or Binary Search |  | 4 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| \*\*\*\*\*\* Not | required to | show | Total | 70 | Other 30 points from Proj 1 first sheet tab |

****

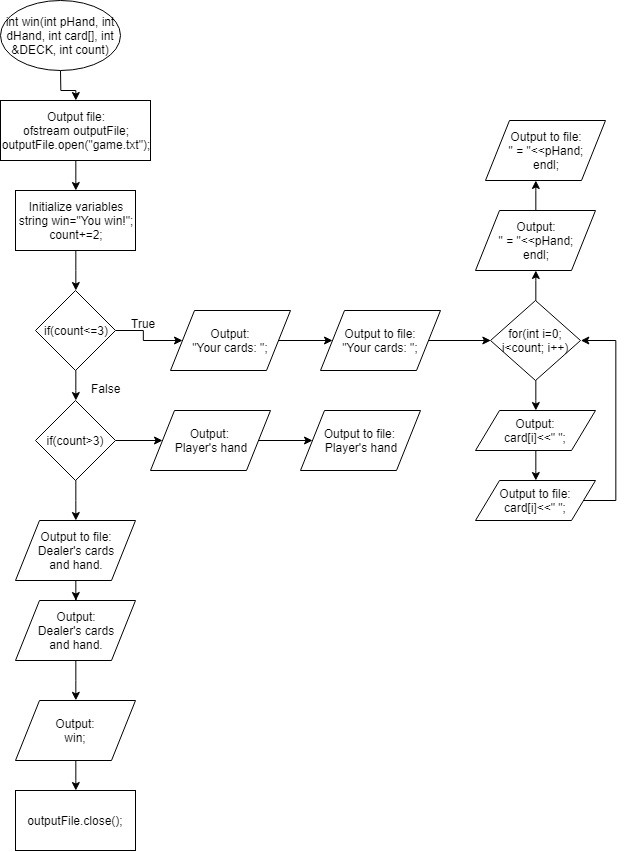
Explanation: The main() function is used to output what the purpose of the game is. Then it asks the player to enter 1 to start the program, or 2 to quit the program. The do loop is used to repeat the switch for input. Case 1 starts the game, case 2 ends the program, and the default asks the user to enter 1 or 2. If the end() function returns false, the program ends.

Explanation: The game function initializes the variables, calls the shuffle function, sets the player’s and dealer’s hands for 2 cards, and shows the player their cards, and the dealer’s cards. The first set of if statements are used to check if the player wins/loses with their first two cards. The while loop is used to control the game. The player is asked if they would like to hit, if they press 1 they hit, if they press 2 they end the game. The for loop is used to output new cards to the player. All the if statements in the while loop check whether or not the player wins, or loses. If the player presses 2, then the program either goes to the win() function, or lose() function. Each if statement sets play to false, because if play is not set to false, then the program will not end. The game will go to the win() function, or lose() function itself if the player reaches a win/loss condition naturally.

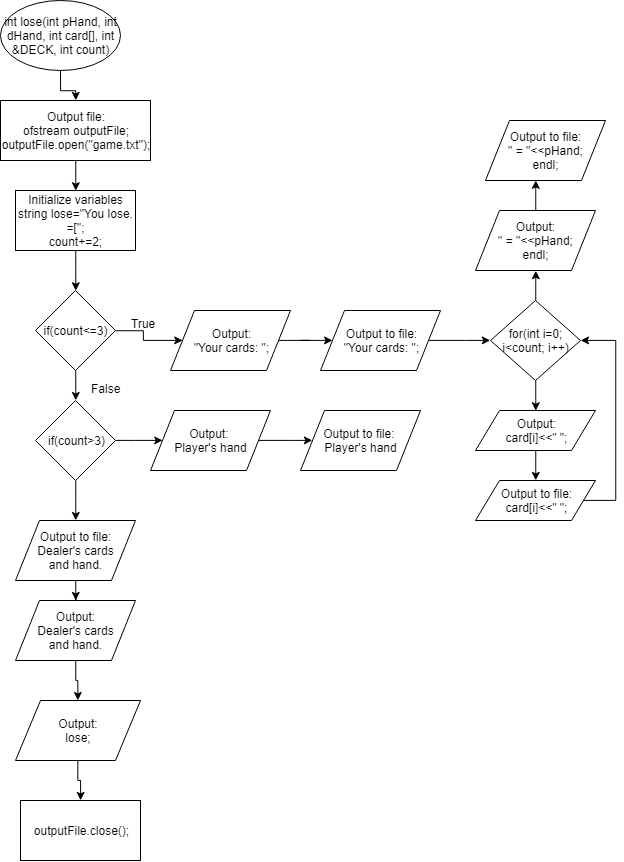
****

****

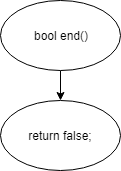
Explanation: I had to make this picture portrait, because it did not fit on the page in landscape.

****

Explanation: The win() function also controls the output file. Basically this function checks if the amount of cards given to the player is less than or greater than 4. If the amount is less than 4, it shows all the cards the player has, and outputs the result to the screen and to the file. If the amount is greater than 4, it only outputs the player’s hand to the screen and to the file. Regardless of what the player’s card count is, this function always shows the dealer’s cards and hand.

****

Explanation: The lose() function is basically the win function, except it outputs a loss instead of a win.

****

Explanation: This function ends the game.